Summer 1991

Parallel-Vector Computation for Geometrically Nonlinear Frame Structural Analysis and Design Sensitivity Analysis

Majdi A. Baddourah
Old Dominion University

Parallel-Vector Computation For Geometrically Nonlinear Frame Structural Analysis and Design Sensitivity Analysis

by

Majdi A. Baddourah
B.S., May 1982, University Of South Carolina
M.S., May 1983, University Of South Carolina

A Dissertation submitted to the faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in
Civil Engineering
Old Dominion University
June 1991

Approved by:

Duc T. Nguyen (Chairman)

Chuh Mei

Gene W. Hou

Leon R.L. Wang

Nahil Sobh

ABSTRACT

Parallel-Vector Computation for Geometrically Nonlinear Frame Structural Analysis and Design Sensitivity Analysis

Majdi A. Baddourah

Old Dominion University

Director: Dr. Duc T. Nguyen

Parallel-vector algorithms are presented for solving the geometrically nonlinear structural problems and obtaining design sensitivity information. A new algorithm is also presented for parallel generation and assembly of the finite element stiffness and mass matrices. The presented assembly algorithm is based on a node-by-node approach rather than the more conventional element-by-element approach. Three different methods, Newton Raphson, Modified Newton Raphson, and the BFGS, are used in the analysis of the nonlinear structural problems. A study is made to determine the performance of each of the mentioned methods in a parallel-vector computer environment. Medium to large-scale, practical problems are solved to evaluate the performance of each method. A hybrid method combining the direct and iterative solvers for linear system of equations is also presented to solve the nonlinear finite element problems. The proposed hybrid method combines the use of the Choleski method and the use of the pre-conditioned conjugate gradient

method, to solve the nonlinear structural problem using the piecewise linear approximation method. A different approach for achieving a parallel-vector speed for the Successive Over Relaxation method is also presented in this work. The new approach for the S.O.R method reduces the cost of communications between processors on shared memory computers. Multi-processor Cray Y-MP and Cray 2 supercomputers are used in this work.

ACKNOWLEDGMENT

To Dr. Nguyen, my Committee chairman, I would like to extend my deepest appreciation for his effort to share his time, knowledge, and enthusiasm. I am especially appreciative of the encouragement and motivation that he has provided throughout my association with him.

To Drs. Chuh Mei, Gene W. Hou, Nahil Sobh, and Leon R.L. Wang, I thank them for their suggestions and support by serving on my Dissertation Committee. In addition, I would like to express my gratitude to the NASA Langley Research Center, the Phillips Air Force Laboratory, and the Civil Engineering Department for providing me the opportunity and the computer facilities to advance my education.

Helpful discussions with T.K. Agarwal and S.A. Eidan (O.D.U doctoral students) is also appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES AND GRAPHS

NOTATION

MATHEMATICAL SYMBOLS

$[\,]$	A rectangular or square matrix.
$[\,]^T$	Matrix transpose.
$[\,]^{-1}$	Matrix inverse.
$\{\,\}$	A column vector.
$\{\,\}^T$	Vector transpose.

LATIN SYMBOLS

A_i	Area of element i.
b	Design variable.
{d}	Direction of the displacement vector.
D.O.F or d.o.f	Degree (or degrees) of freedom.
DSA	Design sensitivity analysis.
E	Modules of elasticity.
E.DSA	Efficiency up for nonlinear design sensitivity analysis.
E.GAS	Efficiency for generating and assembling the $[K_T]$.
E.NON	Efficiency for nonlinear structural analysis.
E.TOT	Efficiency up for total nonlinear structural analysis.
E.TNAS	Efficiency for the nonlinear structural, design sensitivity analysis.

$\{F\}$	Total applied load.
$\{\Delta f\}$ or $\{f\}$	Portion of the applied load.
$[L]$	Lower triangular matrix.
L_j	Length of element j.
$[k]$	Element stiffness matrix.
$[K_L]$,$[K]$	Global linear tangent stiffness matrix.
$[K_T]$	Global linear and nonlinear tangent stiffness matrix.
$[K_{NL}]$	Global nonlinear tangent stiffness matrix.
NP	Number of processors
NLS	Number of loading steps.
$\{Q(U)\}$	Unbalanced loads evaluated at the displacement vector $\{U\}$.
$\{R\}$	Internal resisting loads vector.
s	Step size.
S.DSA	Speed up for nonlinear design sensitivity analysis.
S.GAS	Speed up for generating and assembling the $[K_T]$.
S.NON	Speed up for nonlinear structural analysis.
S.TNAS	Speed up for the nonlinear structural, design sensitivity analysis.
S.TOT	Speed up for total nonlinear structural analysis.
T.BAC	Time spent on back substitution.
T.DSA	Time for nonlinear design sensitivity analysis part.
T.FAC	Time spent on factorizing the global tangent stiffness matrix.
T.GAS	Time spent on generating and assembling the stiffness matrix.
TNI	Total number of iterations to achieve convergence.

T.NON	Time spent on the nonlinear structural analysis part.
T.TOT	Total time spend on nonlinear analysis include read and write.
T.TNAS	Total time spend on nonlinear analysis and design sensitivity.
T.UFO	Total time spend on calculating the unbalanced loads.
$\{U\}$	Displacement vector.

GREEK SYMBOLS

Δ	Small change operator.
$\{\delta\}$	Change in the displacement vector.
$\{\gamma\}$	Change in unbalanced loads.
$\{\psi\}$	Constraints function.
$\{\sigma\}$	Stresses in the element.
λ	Step size in the Arc Length method.
$\{\lambda\}$	Used in DSA to satisfy the adjoint equation.

CHAPTER 1

INTRODUCTION

1.1 Preliminary Remarks

Structural optimization for linear response is a well-defined problem. A large amount of research effort has been devoted to the development of optimal design process for structural systems with linear response. This is mainly due to the fact that many structures have already been designed and used in linear elastic, small-displacement range. Slight nonlinearity for such cases does not violate a linear design basis. However, increasing attention to the use of new materials having nonlinear properties and the design requirements for structures to survive under extreme conditions urge the development of optimal design process with nonlinear finite element structural analysis techniques and remarkable advances in computer technology have made this practical.

To develop an optimal design process for nonlinear structures, it is necessary to develop methods of design sensitivity analysis for nonlinear response. The methods should be efficient, stable and reliable for general applications. The methods are usually related to structural analysis procedures. Some nonlinear structural analysis methods may not be suitable in the overall structural design optimization process. Therefore, development of design sensitivity analysis and hence the optimal design

process for nonlinear systems is possible through a thorough understanding of nonlinear structural analysis procedures.

1.2 Review of Previous Work

In structural mechanics, a problem is nonlinear if the stiffness matrix or the load vector depends on the displacements. Nonlinearity in structures can be classed as material nonlinearity (associated with changes in material properties, as in plasticity) or as geometric nonlinearity (associated with changes in configuration, as in large deflections of a slender elastic beam). In general, for a time-independent problem symbolized as [K] {D} = {R}, in linear analysis both [K] and {R} are regarded as independent of {D}, whereas in nonlinear analysis [K] and/or {R} are regarded as functions of{D}.

Many physical situations present nonlinearities too large to be ignored. Stress-strain relations may be nonlinear in either a time-dependent or a time-independent way. A change in configuration may cause loads to alter their distribution and magnitude or causes gaps to open or close. Thus, we see that nonlinear effects may vary in type and may be mild or severe.

An analyst must understand the physical problem and must be acquainted with various solution strategies. A single strategy will not always work well and may not work at all for some problems. Several attempts may be needed in order to obtain a satisfactory results.

There are various solution methods for solving systems of nonlinear equations. Each method has its own advantage and disadvantage. A single solution method will not always work well and may not work at all for some problems.

The incremental procedure is probably the simplest way to perform a nonlinear finite element analysis. In this procedure, the applied load is divided into several load levels. Within each load level, a linear analysis is performed. Since linear approximations are used, the equilibrium at the next load level will not be exactly satisfied, therefore, iterations within each load step are usually required. Small increments in the load step will generally give accurate results. The computational efforts, however , can be prohibitively high.

The Newton-Raphson (N-R) method Ref. [Bathe,1982], due to its quadratic rate of convergence, is an attractive method for solving a system of nonlinear equations. There is a major drawback, however, associated with the N-R method: the coefficient (tangent stiffness) matrix and the right-hand side vector of the above linear system of equations need to be solved repeatedly, and hence, the N-R iteration process can be quite expensive.

To alleviate the above difficulties, the modified Newton-Raphson (mN-R) method Ref. [Cook,1989] is often used. In the mN-R methods, the original (symmetrical) tangent stiffness matrix can be used throughout the iteration process, thus one only needs to form and factorize the (original, symmetrical) tangent stiffness once. This mN-R methods, however, requires more iterations to converge to the same prescribed tolerance.

The BFGS method Ref. [Matthies an Strang,1979] provides a compromise between the full reformation of the stiffness matrix performed in the full N-R methods and the use of a stiffness matrix from a previous configuration as is done in the mN-R method.

In numerical methods of optimization, one must determine the effect of a change in the current design on the cost and constraint functions, i.e., evaluate the gradients of response quantities with respect to design variables. This is commonly known as design sensitivity analysis (DSA) and can constitute a major task in any structural optimization program. The gradients are also important in their own right as they represent the trends for the structural performance or constraint functions.

In general, there are two ways to compute the design sensitivity information. The most simple and straightforward way is to use the finite-difference approximation. For example,

$$\frac{dg}{db_i} \simeq \frac{g[b_i+h] - g[b_i]}{h} \tag{1.1}$$

where g is a cost or constraint function, b_i is a design variable, and h is a small perturbation in design. There are two serious shortcomings of using this approach for a nonlinear response problem. First, there is an uncertainty in the choice of perturbation, h. An improper choice may cause truncation or condition errors in the computation. Second, when the number of design variables is large, there is an enormous increase in the number of nonlinear finite-element analyses. The procedure, therefore, can be prohibitively expensive.

The other way is to differentiate implicit functions analytically and develop expressions for the gradients. The design sensitivity analysis for linear structural analysis has been well documented. [Khot and Kamat,1983] derive an optimally criterion of uniform strain energy density to minimize the structural weight with a nonlinear buckling constraints. [Kamat and Ruangsilasingha,1984] proposed a design sensitivity formulation for minimization of structural weight subjected tho equality constraints for minimum potential energy as well as the requirement of singular Hessian of the potential energy. [Ryu et al.,1986] have studied the adjoint variable and direct differentiation methods of design sensitivity analysis. They compared the two approaches for linear as well as nonlinear problems. [Wu and Arora,1986] have derived general procedures for design sensitivity analysis for geometric and material nonlinearities.

Design sensitivity analysis will be performed at the last equilibrium stage as proposed by [Rye et al.,1986] and not at each load level as given by [Gopalakruskna and Greimann,1988]. This can be used because in the present work, the nonlinear analysis as well as the design sensitivity analysis, it is assumed that the load applied on the structure is not a function of the displacement.

Nonlinear finite element analysis, by itself, is numerically intensive. The incorporation of the DSA into a nonlinear finite element procedure, therefore, will add tremendous computation burden on the computers. Existing high performance computers (such as the Cray Y-MP, Cray-2 , Convex, IBM-3090, the Alliant) offer new, powerful capabilities for vector and parallel processing. Existing algorithms which have been primarily developed for sequential computers need to be re-

examined and refined in order to take full advantage of these high performance computers. Significant progress have been reported in the literature [Agarwal et al.,1990] for developing parallel-vector algorithms to solve large system of linear equations and eigenvalue problems [Qin and Nguyen,1991]. Parallel numerical algorithms for system of nonlinear equations, nonlinear finite element analysis have recently been reported in the literature [Salama,1981; Law,1985; Zois,1988]. In Ref. [Chen and Sun,1989], parallel computations were implemented during the assembly and solution equation phases. The algorithm for assembly process in Ref. [Chen and Sun,1989], however, is not general. The algorithm will not work if a large number of processor is used and the substructure is small. In Ref. [Carmona et al.,1990], Parallel-vector numerical algorithms of nonlinear equations have been developed to solve system of nonlinear equations. However, only artificial set of system of nonlinear equations has been tested in Ref. [Carmona et al.,1990].

To the author's knowledge, there exists no papers in the literature which focus on the development of parallel-vector algorithms for solving large-scale, practical DSA of nonlinear response.

1.3 Objectives and Scope

The objective of this research are :

a) Review major existing methods for nonlinear structural analysis.

b) Identify and provide a parallel algorithm for the major computational components which are common to most promising solution methods for solving nonlinear structural analysis

equations.

c) incorporate the developed parallel/vector components into promising solution methods such as the N-R, mN-R and BFGS algorithms.

d) Solve practical nonlinear structural analysis problem in order to evaluate the accuracy and speed of the developed parallel-vector nonlinear structural analysis procedures on the Cray 2 and Cray Y-MP supercomputers.

e) Review existing methods for structural DSA of nonlinear response.

f) Propose a parallel-vector procedure for nonlinear DSA.

g) Solve large-scale practical engineering problems to evaluate the accuracy and speed of the developed parallel-vector algorithm for nonlinear DSA.

Since the focus of this dissertation in not developing new finite elements, only 3-D truss and 3-D frame elements are included in the structural applications. Furthermore, only geometrically nonlinear response under static loads are investigated in this research work. Various solution methods for geometrically nonlinear structural analysis are discussed in chapter 2. Parallel-vector computations in various steps of a geometrically nonlinear procedure and its application are presented in chapter 3. The sequential and parallel-vector procedure for nonlinear DSA are given in chapter 4. Large-scale, practical parallel-vector procedures are also demonstrated in chapter 4. Concluding remarks are presented in chapter 5.

CHAPTER 2

GEOMETRICALLY NONLINEAR FINITE ELEMENT ANALYSIS

2.1 Introduction

In linear analysis of structural problems, one needs to solve the linear system of equations [K] {U} = {F}, where [K] is the linear global tangent stiffness matrix, the vector {U} is the displacements of the structure, and the vector {F} is the external applied loads on the structure. In geometrically nonlinear finite element analysis, the same matrix equation applies, [K] {U} = {F}, where [K] is dependent on the displacements, and the load vector {F} can also be dependent on the displacements. The total global tangent stiffness matrix for nonlinear analysis [K] = $[K_T]$ can be written as the sum of $[K_L]$ and $[K_{NL}]$, where $[K_L]$ is the global linear tangent stiffness matrix, and $[K_{NL}]$ is the global nonlinear tangent stiffness matrix. The matrix $[K_{NL}]$ for the truss, and beam elements are given in Ref. [Przemieniecki,1968]. More discussion about the geometrically finite element analysis is given in Ref. [Cook,1989; Bathe,1982]. In the present work, the total Lagrangian formulation and the following assumptions were used for the geometrically nonlinear finite element analysis:

Large displacement, large rotations, small strains.

Linear stress-strain relations for the total response.

The fixed boundary condition do not change during the entire analysis.

Analysis is performed up to the limit point.

Initial displacements and stresses are zero.

The load applied is not a function of the displacement nor the design variable.

Incremnts of the load steps are the same except in the Arc-Length Method.

2.2 Piecewise Linear Approximation Method

In this method, the total applied load vector {F} is divided into several load levels (steps) $\{f_0\}$, $\{f_1\}$, $\{f_2\}$, and so on. For each load level, an approximate linear analysis is performed, and the response is accumulated to form the total response of the structure. In the present work, the total load vector is divided into n equal load levels {f}. In the first load level (iteration) the global linear stiffness matrix $[K_L]$ is generated an assembled to form the linear system of equations:

$$[K_L] \{\Delta U\} = \{f\} \tag{2.1}$$

The above equation is then solved for the incremental displacement $\{\Delta U\}$ and the total displacement vector is given by:

$$\{U\} = \{\Delta U\} \tag{2.2}$$

Using the total displacement vector {U}, the stresses for each element and the unbalanced loads at each node are calculated . The unbalance loads vector $\{\Delta f\}$ is added to the load vector {f} to form the new load vector $\{f+\Delta f\}$, which will be used in the next iteration. Using the incremental displacement vector $\{\Delta U\}$, the coordinates of the nodal points are updated. At any iteration after the first iteration,

the total global stiffness matrix $[K_T]$ is formed and the generated linear system of equations is written as:

$$[K_T] \{\Delta U\} = \{f + \Delta f\} \quad (2.3)$$

The above equation is then solved for the incremental displacement vector $\{\Delta U\}$ and the total displacement vector can be updated as:

$$\{U\} = \{U\} + \{\Delta U\} \quad (2.4)$$

The stress, the unbalance load, and the coordinates of the nodal points are found as in the previous step. The process is repeated for a total number of iteration equal to n.

This method is quite simple, however, the approximate response drifts further from the exact solution with every load level. The drifts can be reduced if the total load is divided into many small load steps, which will increase the time to complete the analysis.

2.3 Newton-Raphson (N-R) Method

In Newton-Raphson method the equilibrium equation is written as:

$$F(U) - R(U) = 0 \quad (2.5)$$

where F(U) is the total applied external load vector, and R(U) is the total internal resisting load vector. In order to satisfy the equilibrium equations, the internal loads must be equal to the external loads applied on the structure. Using Taylor series, the above equation can be expanded and simplified as:

$$\left[\frac{\partial R}{\partial U}\right](\Delta U)=F-R \tag{2.6}$$

where F - R is the unbalanced loads at each step, and [∂R / ∂U] is the tangent stiffness matrix [K_T]. More discussion on this subject is given by Ref. [Bathe,1982]. In the present work, the total applied load vector {F} is divided into n equal load vectors {f}. In the first iteration of the first load level, the analysis is started by generating and assembling the global linear tangent stiffness matrix [K_L]. The generated linear system of equations is written as:

$$[K_L]\ \{\Delta U\} = \{f\} \tag{2.7}$$

The above equation is then solved for the incremental displacement vector {ΔU}, and the total displacement vector {U} can be written as:

$$\{U\} = \{\Delta U\} \tag{2.8}$$

Then, using the total displacement vector {U}, the stresses for each element, and the unbalanced loads {Δf} at each node can be calculated. The unbalanced load vector {Δf} is used in the next iteration. Using the incremental displacement vector {ΔU}, the nodal coordinates are updated. In the second iteration of the first load level the total global tangent stiffness matrix (linear and nonlinear) [K_T] is generated and assembled, and the generated system of linear equations is given by:

$$[K_T]\ \{\Delta U\} = \{\Delta f\} \tag{2.9}$$

Solving the above equation for {ΔU} and updating the total displacement vector {U} as follows:

$$\{U\} = \{U\} + \{\Delta U\} \tag{2.10}$$

The stresses, unbalanced loads, and the nodal coordinates are then calculated as in the previous step. The above procedure is iterated until, convergent criteria is achieved or the maximum number of iterations is reached. In order to achieve convergence, the norm of the unbalanced load vector $\| \{\Delta f\} \|_2$ must be within a preset tolerance. Achieving convergence in the first (or any load) level is commonly known as completing one equilibrium iteration. In the first iteration of the second load level, one starts generating and assembling the total global tangent stiffness matrix $[K_T]$. The last unbalanced load vector $\{\Delta f\}$, obtained at the completion of the previous equilibrium iteration, is added to the load vector $\{f\}$ to give a new load vector $\{f+\Delta f\}$. The generated linear system of equation can be written as:

$$[K_T] \{\Delta U\} = \{\Delta f+f\} \tag{2.11}$$

The procedure is then continued as in the first load level. To complete the analysis, n number of equilibrium iterations are used.

In N-R method, a new set of linear system of equations must be generated and factorized many times for each load level. The number of computations in this procedure is expensive. N-R method can be a very reliable method in solving nonlinear structural problems when an initial starting vector is reasonably good. The convergence rate of the method is quadratic.

2.4 Modified Newton-Raphson (mN-R) Method

Modified Newton-Raphson method follows the same derivation as the N-R method. In mN-R method, the global tangent stiffness matrix is generated once, and the linear system of equations are factorized once, for one equilibrium iteration. In

the present work, the total applied load is divided into n equal load levels {f}. At the first iteration of the first load level the global tangent stiffness matrix $[K_T]$ is generated and assembled. The generated linear system of equations is written as:

$$[K_L] \{\Delta U\} = \{f\} \tag{2.12}$$

Then equation is solved for the incremental displacement vector $\{\Delta U\}$, and the total displacement vector can be written as:

$$\{U\} = \{\Delta U\} \tag{2.13}$$

Then the stresses, the unbalanced loads vector $\{\Delta f\}$, and the nodal points coordinates are updated just as in N-R method. At the second iteration of the first load level a new tangent stiffness matrix is generate using the $[K_{NL}]$ and $[K_L]$ to form the $[K_T]$. The generated linear system of equations are then solved for the displacement vector $\{\Delta U\}$ as follows:

$$[K_T] \{\Delta U\} = \{\Delta f\} \tag{2.14}$$

The total displacement vector can be updated as:

$$\{U\} = \{U\} + \{\Delta U\} \tag{2.15}$$

At the next iteration the algorithm does not require the generation and factorization of a new global tangent stiffness matrix; instead, the same factorized linear system of equations are used to obtain the new vector $\{\Delta U\}$. The process is repeated until we achieve the norm of unbalanced load vector$\| \{\Delta f\} \|_2$ to be within a preset tolerance. At second or any equilibrium iteration greater than the first equilibrium iteration, a new total tangent stiffness matrix is generated and factorized once. From that point the procedure is repeated as in the first equilibrium iteration. The number

of equilibrium iteration performed should equal to n to complete the nonlinear analysis.

For the first equilibrium iteration, the generation of the tangent stiffness matrix twice, will help in the convergence rate since the second tangent stiffness matrix contains the nonlinear terms. For the second equilibrium iteration onward, generate, and factorize the total global stiffness matrix is generated and factorized only once at the beginning of the equilibrium iteration. This will reduce the computer time for generating and factorizing the total global tangent stiffness matrix. Due to the above discussion, it is observed that more iterations are needed in order to achieve the same convergence as in N-R method.

2.5 BFGS Method

Broyden-Fletcher-Goldfarb-Shanno (BFGS) method Ref. [Bathe,1982] is also used to solve the geometrically nonlinear finite element problem in this work. This method is classified under a group of methods called quasi-Newton methods. The general idea behind these methods is to update the global tangent stiffness matrix without the need of generating and factorizing the global tangent stiffness matrix at each load level. This method is well explained in Ref. [Matthies and Strang,1979]. In the present work, the total load vector {F} is divided into equal load vectors {f}. In the first load level, the linear global tangent stiffness matrix $[K_L]$ is generated. The generated linear of equation is solved for $\{d_o\}$

$$[K_L]\ \{d_o\} = \{f\} \qquad \textbf{(2.16)}$$

where $\{d_o\}$ is the direction of travel in the space, the subscript denotes the iteration number. $[K_L]$ can be written as $[L][L]^T$. The secant method Ref. [Chapra and Canale,1985] is used to solve the following scalar function:

$$G(s_o) = \{d_o\}^T Q(s^*\{d_o\}) = 0.0 \tag{2.17}$$

where $Q(s_o^*\{d_o\})$ is the unbalanced loads evaluated at the displacement vector $(s_o^*\{d_o\})$ and s_o is the step size. Equation (2.17) express minimization of the energy error. The total displacement vector can be updated as

$$\{U_1\} = \{U_o\} - (s_o^*\{d_o\}) \tag{2.18}$$

In the second step the new value of the vector $\{d\}$ can be written as:

$$\{d_1\} = (I+\{w_1\}\{v_1\}^T)\,[K_L]^{-1}\,(I+\{v_1\}\{w_1\}^T)\,Q(U_1) \tag{2.19}$$

In the above equation it appears that the inverse of $[K_L]$ should be computed, where by following the next steps finding the inverse of $[K_L]$ can be avoided.

$$\text{Solve } \{b_1\} = \{Q(U_1)\} + (\{(w_1)^T\}\,\{Q(U_1)\})\{v_1\} \tag{2.20}$$

$$\text{Use } [L][L]^T\{c\} = \{b1\} \quad \text{where } \{c\} = [K_T]^{-1}\{b1\} \tag{2.21}$$

$$\text{The new vector } \{d_1\} = \{c\} + (\,(\{v_1\}^T\{c\})\,\{w_1\}\,) \tag{2.22}$$

Update the new vector $\{U\}$; in general, the new vector $\{U\}$ can be written as:

$$\{U_i\} = \{U_{i-1}\} - (s_{i-1}^*\{d_{i-1}\}) \tag{2.23}$$

The new vector $\{d\}$ can be updated according the following equation:

$$\{d_i\}=(I+\{w_i\}\{v_i\}^T)(I+\{w_{i-1}\}\{v_{i-1}\}) \ \ldots \ [K_T]^{-1} \ \ldots \ (I+\{v_{i-1}\}\{w_{i-1}\}^T)(I+\{v_i\}\{w_i\}\{Q(U_i)\} \tag{2.24}$$

where

$$\{v_i\}=\{Q(U_{i-1})\}\left(1+s_{i-1}\left(\frac{\{d_{i-1}\}^T\{\gamma_i\}}{\{\delta_i\}^T\{Q(U_{i-1})\}}\right)^{1/2}\right)-\{Q(U_i)\} \tag{2.25}$$

and

$$w_i=\frac{1.0}{\{\delta_i\}^T\{\gamma_i\}}\{\delta_i\} \tag{2.26}$$

where

$$\{\delta_i\}=\{U_i\}-\{U_{i-1}\} \tag{2.27}$$

and

$$\{\gamma_i\}=\{Q(U_i)\}-\{Q(U_{i-1})\} \tag{2.28}$$

The above procedure is repeated until the $\| \{Q(U)\} \|_2$ is within preset tolerance or the maximum number of iterations is exceeded. In the computer program generated for this work, the maximum number of iteration i is set to a maximum value equal to eight. This is important since the time needed to calculate the vector $\{d_i\}$, where i is greater than eight, exceeds the time needed to assemble, factorize and solve a new global tangent stiffness matrix. For the above reason, a new tangent stiffness matrix $[K_T]$ (which include the nonlinear part) is generated and factorized, and used as a new starting global tangent stiffness matrix, after eight iterations, in the BFGS method.

The BFGS has linear convergence characteristic. The method is considered to be stable in nonlinear structural analysis. In general, this method will converge faster than the m.N-R method.

2.6 Arc-Length Method

In the present work, the arc length method is used in order to find the maximum proportional load the structure can carry before instability occurs. If one of the methods discussed above produced a negative definite global tangent stiffness matrix, or the method diverges, then the analysis will shift to the use of the Arc-Length method. The method is discussed in details in Ref. [Riks,1979]. The method can be summarized by the need to find the value of $\Delta\lambda$ which we will multiply the vector {f} in order to achieve convergence and the maximum load vector {F} the structure can sustain, where $\Delta\lambda^{(j)}$ is given by:

$$\Delta\lambda^{(j)} = -\frac{\{\Delta U^{(1)}\}^T \cdot \{\Delta U^{(j)}\}^{II}}{\{\Delta U^{(1)}\}^T \cdot \{\Delta U^{(j)}\}^{I} + \Delta\lambda^{(1)}} \qquad (2.29)$$

where $\{\Delta U^{(j)}\}^I$ can be determined by using the last $[K_T]$ at equilibrium and solving:

$$[K_T]\ \{\Delta U^{(j)}\}^I = \{f\} \qquad (2.30)$$

$\{\Delta U^{(j)}\}^{II}$ can be determent from the solving the equation:

$$[K_T]\ \{\Delta U^{(j)}\}^{II} = \{\Delta f\} \qquad (2.31)$$

Where $\{\Delta f\}$ is the unbalanced loads after the last equilibrium stage. $\Delta\lambda^{(1)}$ can be suppressed for many degrees of freedom system, so the vector $\{\Delta U^{(1)}\}$.

2.7 Convergence Criteria

In order to complete the nonlinear structural analysis, efficient criteria for checking the numerical convergence of the problem is critically needed. At each iteration, using N-R, mN-R, BFGS, and the Arc-Length, checking is required to know if the problem is converging or diverging. There are many ways to check the convergence criteria as given in Ref. [Bathe,1982]. One of the goals of the present study is to compare the total time used to complete the nonlinear structural analysis for different methods. In order to do that, the unbalanced loads are used as a convergence criteria. This can be show as:

$$\|\{Q^i\}\|_2 \leq \epsilon \tag{2.32}$$

where $\{Q^i\}$ is the unbalanced load vector at any iteration i, vector $\{f\}$ is the applied loads at any load step, and ϵ is any preset tolerance value.

CHAPTER 3

PARALLEL-VECTOR COMPUTATION FOR GEOMETRICALLY NONLINEAR ANALYSIS

3.1 Parallel Fortran Language

FORCE is a preprocessor which produces executable parallel code from a combination of Fortran and a set of simple, yet portable, parallel extensions tailored to run efficiently on parallel computers[Jordan at al.,1985]. Force provides a machine independent tool for parallel programming. With certain exceptions, FORCE includes all Fortran constructs. The parallel extensions include such construction as Pre- and Self-scheduled Do loops, Barrier, Private and shared Variables, Produce and Consume. We will only discuss some of these constructs which we have used in our code. Barrier will allow only one processor to execute the block of statements in between the two statements while the rest of the processors are waiting. Critical and End Critical will allow each processor to execute the block of executable statements in between the statements in a sequential manner. Presched Do will execute the do loop in parallel, that is all the processors will be working at the same time for the do variable. Also, the variables can be either shared between all the processors or private (each processors will have its own value for the same variable name). The particular manner in which these constructs are

implemented on various parallel computers is hidden from the users (i.e users can write efficient, yet portable, parallel code with no need to refer to esoteric details on multi-tasking or parallel programming contained in specific vendor manuals, knowledgeable computer scientists did this when FORCE was first installed). Programs written in Force are easily ported to other parallel computers with no change provided FORCE is running on the target computer. In simple definition, we can define the speed up of the algorithm by:

$$\text{Speed up} = \frac{\text{Total run time for one processor}}{\text{total run time for multi-processors}} \tag{3.1}$$

The efficiency of an algorithm can be given as:

$$\text{Efficiency} = \frac{\text{Speed Up}}{\text{Number of processors}} * 100 \tag{3.2}$$

3.2 Common Computational Steps In Existing Geometrically Nonlinear Structural Analysis

A closer look to the N-R, mN-R, and BFGS methods will indicate the following common computational steps involved.

Step 1: Guess the initial solution, using the linear tangent stiffness matrix.

Step 2: Compute the tangent stiffness matrix $[k_T(U)]$.

Step 3: Compute the displacement vector $\{\Delta U\}$, by solving the linear system of equations.

Step 4: Update the displacement vector and coordinates.

Step 5: Compute the unbalanced loads, and update the load vector.

Step 6: Convergence check.

The computational involved in step one and six are so trivial and insignificant. For this reason only steps two through five of the above general procedure need to be implemented in a parallel-vector computer environments.

In this chapter, each of the above steps will be discussed in a parallel-vector computer environment. The results of some of the above steps will be discussed later with the provided examples.

3.3 Parallel Generation and Assembly of Element Stiffness and Mass Matrices

Since the time for solving a system os linear equations has been reduced significantly by using the recently developed parallel-vector equation solver [Agarwal,1990], generating and assembling the structural matrices now represent a significant portion of the total CPU time [Storaasli et al.,1990], for many practical applications. This is especially true for nonlinear structural analysis, structural optimization, and control structure interaction, since in these applications new element matrices need to be generated and assembled repeatedly in an iterative procedure. In this section, a procedure for the new approach for generating and assembling the element stiffness matrices is given. The results for this procedure will be given later using practical, structural examples.

Conventional Element-By-Element algorithm

In this conventional procedure, the element stiffness (or mass) matrices are generated and their contributions to the structural (global) stiffness (or mass) matrix can be obtained as shown in the following pseudo-Fortran statements:

DO 1 e = 1 , number of elements

. generate element stiffness matrix $[k^{(e)}]$.

. add contribution of $[k^{(e)}]$ to structural matrix

$$[K] = \Sigma[k^{(e)}]$$

1 Continue

In parallel computer environment, however, synchronization is a problem. This may be demonstrated by the simple 3-bar truss of Figure 3.1. In the above procedure, if each e^{th} finite element is assigned to a separate processor, then at node one (see Figure 3.1), for example, both element two and three will often need to write simultaneously their element stiffness contribution to the structural stiffness matrix at the same locations.

This synchronization problem can be partially overcome by either setting the locks in common shared memory pool or by special numbering of the elements throughout the entire structure [Chen and Sun,1985]. The first method [Chen and Sun,1985] (setting the local locks) reduces the speed of parallel assembly considerably. The second method, special numbering of the elements, is not general since the method will not work if a large number of processors are used and the substructure is small.

To alleviate the above synchronization problem a node-by-node approach is prevented in the next subsection.

Node-By-Node Algorithm

In this new algorithm, element matrices will be generated and assembled in a node-by-node fashion. For a two-node (node i and node j) truss element, for example, a two dimensional, four by four element stiffness matrix [k(e)] can be symbolically represented as:

$$[\, K^{(e)} \,] = \begin{bmatrix} k_{ii}^{(e)} & k_{ij}^{(e)} \\ k_{ji}^{(e)} & k_{jj}^{(e)} \end{bmatrix} \tag{3.3}$$

In equation 3.3 $k_{ii}^{(e)}$ and $k_{jj}^{(e)}$ refer to the two by two sub-matrices which represent a portion of an element stiffness matrix attached to node i and node j, respectively. The coupling effect between nodes i and j of an element stiffness matrix $[k^{(e)}]$ is represented by a two by two sub-matrix $k_{ij}^{(e)}$ (or the transpose of the two by two sub matrix $k_{ij}^{(e)}$). Using the three-bar truss structure (see Figure 3.1) as a simple illustration, the new algorithm can be described in the following step-by-step procedure:

step 1 Element Connectivity Data

The standard element connectivity data of the three-bar truss structure (see Figure 3.1) can be readily obtained as shown in Table 3.1

Step 2 Node Connectivity Information

In this step, elements which are attached to nodes i and nodes j of the entire structure can be readily identified. For the three-bar truss structure shown in Figure 3.1, the node connectivity information can be generated as shown in Table 3.2.

In Table 3.2, element two, for example, appears in the last two columns because element two is connected by node i equal to one and node j equal to three. Similarly, element one also appears in the last two columns because element one is connected by node i equal to two and node j equal to three. This step is an additional overhead cost of the proposed new algorithm, since the information generated in this step is usually not required in conventional finite element codes.

Step 3 Parallel Generation and Assembly of $k_{ii}^{(e)}$ for Each Node of a Structure.

In this step, the portion $K_{ii} = \Sigma\ k_{ii}^{(e)}$ of the structural (or global) stiffness (or mass) matrix K is generated and assembled in a parallel computer environment. From the nodal connectivity information generated in the previous step, each node can be assigned to a separate processor. Thus, in the three-bar truss structure (see Figure 3.1), node one will be assigned to processor one, therefore will generate $k_{ii}^{(e)}$ of the structural stiffness matrix K. Simultaneously, node two is assigned to processor two which generate $k_{ii}^{(e)}$ of elements e equal to one and three in a sequential fashion and add its contribution to appropriate locations (D.O.F three and four) of the structural stiffness matrix K. At the same time, processor three is assigned to node three (associated with D.O.F five and six). In this step, processor three is idle, as there are no elements with node i equal to three (see the third column of Table 3.3)

The parallel generation/assembly of $k_{ii}^{(e)}$ for each structural node can be represented as in Table 3.3. In Table 3.3, the rows and columns represent the location in the global matrix, and the entries represent the contributions from

different element numbers. As can be seen from Table 3.3, this step can be done entirely in parallel with no communication between processors. From Table 3.3, there is no overlapping between processor one (which is assigned to D.O.F. one and two), processor two (which is assigned to D.O.F. three and four), and processor three (which is assigned to D.O.F. five and six).

Step 4 Parallel Generation and Assembly of $k_{jj}^{(e)}$ for Each Node of a Structure.

In this step, the portion $K_{jj} = \Sigma\ k_{jj}^{(e)}$ of structural stiffness matrix K is generated and assembled in a parallel computer environment. Each node is again assigned to a separate processor, and the information in the last column of Table 3.4 is used here. Processor one, which is assigned to node one, will generate $k_{jj}^{(e)}$ of element 3, and add its contribution to appropriate locations (D.O.F. 1 and 2) of the structural stiffness matrix K. Simultaneously, processor three, which is assigned to node three, will generate $k_{jj}^{(e)}$ of elements e equal to one and two, and add its contribution to appropriate locations (D.O.F. five and six) of the structural stiffness matrix K. At the same time, processor two is assigned to node two (associated with D.O.F. three and four). In this step, processor two is idle, since there are no elements with node j equal to two. The parallel generation/assembly of $k_{jj}^{(e)}$ for each structural node is shown in Table 3.4.

Step 5 Parallel Generation and Assembly of $k_{ij}^{(e)}$ for each Node of a Structure.

In this step, the portion $K_{ij}^{(e)}$ of the structural stiffness matrix, K, is generated and assembled in a parallel computer environment. To find out what

elements are attached to a given node of a structure, information on either nodes i (see the 3rd column of Table 3.2) or nodes j (see the 4th column of Table 3.2) can be used to generate the portion K_{ij} of the structural stiffness matrix K. In this study, the information for nodes j is used in this step. Thus, processor one is assigned to node one to process element number three. Element three is connected to D.O.F. one, two, three, and four and its contribution to K_{ii} and K_{jj} have been done in step three and step four, respectively. Processor one will generate $k_{ij}^{(e=3)}$ and add its contribution to the appropriate locations of K_{ij}. Simultaneously, processor three is assigned to node three to process elements one and two. Processor three will, therefore, generate $k_{ij}^{(e)}$ for elements e equal to one and two, and add its contribution to the appropriate locations of k_{ij}.

In this step, processor two is idle since there are no elements with node j equal to two. The parallel generation/assembly of $k_{ij}^{(e)}$ for each structural node is conveniently represented in Table 3.5.

Since the structural stiffness matrix K is symmetric, only the upper-half of the matrix is considered.

The above five-step procedure to generate and assemble element stiffness matrices in parallel is quite general since there is no assumption on the type of element used in the finite element model. For a more convenient and efficient computer implementation, the execution in step 5 should be included in step 3. Thus, the overhead cost due to the recalculation of some parameters for generating the element stiffness matrix can be reduced.

It should be emphasized here that practical finite element models have a large number of degrees-of-freedom. For these large models, many finite elements need to be processed. Furthermore, there are fewer processors available as compared to the number of elements in a large finite element model. Thus, a balance of the work load among the too many processors is well preserved. In an ideal parallel algorithm, each processor is expected to process the same number of finite elements. Thus the idle time of the processors is reduced through this assembly process.

3.4 Parallel-Vector Linear Equation Solver

3.4.1 Choleski Method

The basic Choleski algorithm can be described by decomposing the following linear system of equations :

$$[K]\ \{U\} = \{F\} \tag{3.4}$$

into

$$[L]\ [L]^T\ \{U\} = \{F\} \tag{3.5}$$

where [L] is a lower triangular matrix. In this dissertation, the decomposition and the back substitution of the stiffness matrix is performed using the subroutine developed in Ref. [Agarwal et al.,1990]. This subroutine is considered to be one of the fastest direct equation solver using the Choleski method. The stiffness matrix is stored in the one dimensional array, according to the row by row fashion.

In the present work, however, the stiffness matrix is presented by the two dimensional array in order to simplify the discussion.

3.4.2 Successive Over Relaxation Method

The linear system of equations [K] {U} = {F} can be solved using the Successive Over Relaxation (S.O.R.) method, where the N by N matrix [K] can be written as:

$$[K] = [D] + [L] + [L]^T \tag{3.6}$$

[L] and $[L]^T$ are the strictly lower and strictly upper triangular part of the matrix [K], and [D] is the diagonal matrix. The above equation can be written in an iterative form as:

$$[D]\{U\}^{m+1} + \omega\,[L]\{U\}^{m+1} = (1 - \omega)[D]\{U\}^m - \omega\,[L]^T\{U\}^m + \omega\,\{F\} \tag{3.7}$$

where the superscript m indicates the iteration number, and ω is an acceleration factor. From the above equation it can be observed that matrix by vector multiplications are needed to in the S.O.R. algorithm. Since the stiffness matrix [K] is stored in a row format, a different scheme is used to achieve parallel-vector matrix by vector multiplication. The procedure is initiated by multiplying the [L] by {U} in parallel with loop unrolling level two as follows:

```
      Presched do 20 I = 2 , (N/2) * 2 , 2
      Z(I) = Z(I) + L(I,I-1) * U(I-1)
      DO 10 J = I+1 , MAXBAND
      Z(J) = Z(J) + L(J,I) * U(I-1) + L(J,I+1) * U(I)
10    CONTINUE
20    End presched do
```

since we have loop unrolling level 2 the remainder of the matrix vector multiplication is given by:

```
      Presched do 40 I = (N/2) * 2 +1 , N
      DO 30 J = I , MAXBAND
      Z(J) = Z(J) + L(J,I) * U(I-1) + L(J,I+1) * U(I)
30    CONTINUE
40    End presched do
```

In which, the vector {Z} is an extra array and should be declared as the private variable. The vector {T}, to be defined in the following Fortran statements, is an extra array used for multiplication purpose and should be declared as the shared variable. Using Critical block, each processor will sequentially update the vector {T} using its own value of the vector {Z} as follows:

```
      Critical TYPE1
      DO 50 I = 1 , N
      T(I) = T(I) - Z(I)
50    CONTINUE
      End critical
```

The algorithm then continues solving equation (3.7) and updating the last value in the vector {U}, by using all the old values of the vector {U}.

```
      U(N) = ( ( 1 - ω ) * D(N,N) * U(N) + ω * ( F(N) -T(N)) ) / D(N,N)
```

Next, update the rest of the vector {U} in parallel as follow:

```
      DO 70 I = N-1 , 1 , -1
      SUM = F(I) -T(I)
      DO 60 J = I+1 , N
      SUM = SUM - U(I) * U(I,J)
60    CONTINUE
      U(I) = ( ( 1 - ω ) D(I,I) * U(I) + ( ω *  SUM ) ) / D(I,I)
70    CONTINUE
```

It should be noted that since the above statements are executed in a shared memory type computers, the available values of U(I) will be used. Since there are no communications in the previous steps, the value of U(I) whether it is updated or not updated, will be shared by other processors concurrently. The S.O.R. algorithm is considered to be converged when the following "error norm" is small:

$$\| [K] \{U\} - \{F\} \|_2 \approx \text{a preset tolerance} \qquad (3.8)$$

3.4.3 Pre-conditioned Conjugate Gradient Method (P.C.G)

The pre-conditioned conjugate gradient method [Ortega,1988] can be written in a parallel-vector environment. In this section the basic P.C.G is given, and some of the steps that can be executed in a parallel-vector environment is identified. The linear system of equations $[K_T] \{U\} = \{F\}$ needed to be solved using the P.C.G

method. The basic P.C.G algorithm can be given in the next following pseudo-Fortran statements:

Choose $\{U^1\}$ as an initial guess.

Set $\{RES^1\} = \{F\} - [K^T]\{U^1\}$, where $\{RES^1\}$ is the residual.

Solve $[M]\{RBAR^1\} = \{RES^1\}$, where [M] is a preconditioning matrix.

Set $\{P^1\} = \{RBAR^1\}$

For m = 1, 2, ...

$$\alpha_m = -(\{RBAR^m\},\{RES^m\}) / (\{P^m\},[K_T]\{P^m\}) \quad (3.9)$$

$$\{U^{m+1}\} = \{U^m\} + \alpha_m \{P^m\} \quad (3.10)$$

$$\{RES^{m+1}\} = \{RES^m\} + \alpha_m [K_T] \{P^m\} \quad (3.11)$$

Test for convergence, α_m should be within a preset tolerance.

$$\text{Solve } [M]\{RBAR^{m+1}\} = \{RES^{m+1}\} \quad (3.12)$$

$$\beta_m = \{RBAR^{m+1}\},\{RES^{m+1}\} /\{RBAR^m\},\{RES^m\} \quad (3.13)$$

$$\{P^{m+1}\} = \{RBAR^{m+1}\} + \beta_m \{P^m\} \quad (3.14)$$

To find $\{RES_1\}$, matrix by vector multiplication is needed. The step can be executed in parallel with vector speed, using loop unrolling level two is given by following the next statements:

```
Presched do 20 I = 1 , N
SUM = 0.0
DO 20 J = I , MAXBAND
```

```
      SUM = SUM + K(I,J) * U(I)
10    CONTINUE
      RES(I) = F(I) - SUM
20    End Presched do
      Presched do 40 I = 2 , (N/2) * 2 , 2
      Z(I) = Z(I) + K(I,I-1) * U(I-1)
      DO 30 J = I + 1 , MAXBAND
      Z(J) = Z(J) + K(J,I) * U(I-1) + K(J,I+1) * U(I)
30    CONTINUE
40    End presched do
      Critical TYPE1
      DO 50 I = 1 , N
      RES(I) = RES(I) - Z(I)
50    CONTINUE
      End critical
```

We must declare the variable SUM as Private variable. Each processor will have its own value of the variable SUM. The vector {Z} is an extra array and is declared as private variable.

Solving [M] $\{RBAR^1\} = \{R^1\}$ is performed sequentially using the solver subroutine in Ref. [Agarwal,1990].

Setting the vector $\{P^1\} = \{RBAR^1\}$, can be executed concurrently using Presched do command.

To find α_m in equation (3.9), it involves multiplying the matrix $[K_T]$ by the vector $\{P^m\}$, and can be implemented as discussed earlier. It also involves multiplying the two vectors ($\{RBAR^m\}$,$\{RES^m\}$) and ($\{P^m\}$,$[K_T]\{P^m\}$). The procedure to multiply two vectors together is given by:

```
      Presched do 60 I = 1 , N
      SUMP = SUMP + RBAR(I) * RES(I)
      SUMP2 = SUMP2 + P(I) * T(I)
60    End presched do
      Critical TYPE2
      SUM = SUM + SUMP
      SUM2 = SUM2 + SUMP2
      End critical
```

where SUMP, and SUMP2 are declared as private variables. SUM, and SUM2 are declared as shared variables. The vector {T} is the resulting product of the matrix $[K_T]$ by the vector $\{P^m\}$. Then α^m can be computed sequentially as follows:

```
      Barrier
      ALFA = SUM / SUM2
      End barrier
```

Equations (3.10) through equation (3.14) involves in a matrix by vector multiplications or a vector by vector multiplication. One can easily follow the steps given earlier for the completing the algorithm.

3.4.4 Hybrid Direct-Iterative Method

In this section the implementation of the hybrid direct-iterative solver for the linear system of equations for the analysis of nonlinear finite element problems is given. The piecewise linear approximation method is used for solution of the nonlinear structural analysis problem. In general, iterative methods will have a rapid convergence if a good initial guess was chosen. For the above reason, it is suggested to use Choleski method for the first and the second iteration for solving the generated linear system of equations in the piecewise linear approximation method. After the second iteration, the displacement vector $\{\Delta U\}$ can be used as a good initial guess to any iterative method for solving the linear system of equations. It is suggested that two iterations are completed using Choleski method before switching to an iterative method, because in the first iteration only the $[K_L]$ is generated and in the second iteration the nonlinear part start appearing, which gives a better guess for $\{\Delta U\}$. From the above reasons, it is assumed that the linear system of equations at the second and the third , or any two consecutive, iterations will not have a considerable change. Also, the solution of the system of equation $\{\Delta U\}$ will not vary considerably between any two consecutive load steps. The above discussion is valid for the use of S.O.R. or conjugate gradient method. The preconditioned conjugate gradient method requires the use of preconditioning matrix [M] as discussed earlier

in previous section. The factorized matrix $[L][L]^T$ obtained in the second iteration by using Choleski is used as a preconditioning matrix [M] for the next few iterations. After few iterations, one can expect the preconditioning matrix $[L][L]^T$ will not be a good approximation for the current matrix $[K_T]$, because of the nonlinear nature of the problem. To reduce the time spent by using the P.C.G method for many iterations, the matrix is factorized once by Choleski method and a new preconditioning matrix $[L][L]_i^T$ is obtained at iteration i. In order to achieve the above in a systematic manner, the time spent on solving the system of equations by the direct and indirect methods is monitored. In this instance the time used by the iterative solver exceeds the time used in the direct solver, a new factorization of the system of equations is performed. The above discussion is only valid by using one processor. In parallel environment, the preconditioning matrix $[L][L]_i^T$ may still be a good approximation for the matrix $[K_T]$. When to switch the algorithms, however, depend on the parallel speed up of each method. It is important to notice that the speed up for Choleski and the P.C.G will not be the same, in general.

3.4.5 Numerical Performance of Hybrid Direct-Iterative Method.

The results discussed in this section is based on using three dimensional truss with (1872 D.O.F) as shown in Figure 3.1. The results are given in Tables [3.6,3.7,3.8]. The method used for the nonlinear analysis is linear piecewise, and the iterative method used for solving the linear system of equations is P.C.G.

From Table 3.6, using one processor, one can observe that the total time for the nonlinear analysis using the hybrid method (8.33 seconds) is less than the total

time produced by the use Choleski method alone (11.62 seconds). As the number of processors increases, the total time for both methods get closer. The reason for that is because P.C.G. method does not give a good parallel speed. It is also worth mentioning that by using only one processor in the hybrid method, the number of iterations of the P.C.G. increases as the number of the piecewise linear approximation method iterations increases. The behavior is expected since the preconditioning matrix $[L][L]^T$ is not updated and does not represent a good precondition as the number of iterations increases. Since we do not achieve a good parallel speed up by the P.C.G. The use of S.O.R. in parallel vector environment was also investigated. From Table 3.8, one can see that the time spent using the hybrid approach at each iteration is greater or equal to the time spend by the Choleski method alone, where as the total nonlinear analysis time for the hybrid method is less than the total time spent by using only Choleski. The reason is because in the code, the time measured by using Choleski only includes the factorization time and does not include the back-substitution time.

The S.O.R. method can also be used with the Choleski method to solve the nonlinear structural problem. However, in this work, the S.O.R. method is not compared with the Choleski method, because the value of ω is not known. Only the speed up for the S.O.R. method will be discussed in this section for an assumed ω = 1. From Table 3.9, the efficiency decreases as the number of processors increases. Note also the number of iterations increases in the S.O.R. method as the number of processors increase. It is clear that the processors will use any old or updated value of the unknowns. If many of the old values were used, the number of S.O.R.

iterations increase, since the S.O.R. algorithm does not use the updated values (hence, is equivalent to the Gauss-Jacobi algorithm) of the unknowns at one iteration, requires more iterations than the Gauss-Seidel method. In general the speed ups obtained are good compared to the P.C.G method.

3.5 Parallel Updating of the Displacements and the coordinates.

The displacements and the coordinates can be updated with no communications involved at the node level. This can be shown in the following Fortran statements:

```
      Presched do 10 I = 1 , Number_of_Nodes
      DO 20 J = 1 , 6 (Six degrees of freedom at each node)
      U(J) = U(J) + ΔU(J)
20    CONTINUE
      X(I) = X(I) + ΔU(1)
      Y(I) = Y(I) + ΔU(2)
      Z(I) = Z(I) + ΔU(3)
10    End presched do
```

Where X,Y,Z are the coordinates of the nodal points. It is clear that the computation involved in the above statements are not that high, but in nonlinear analysis the process is repeated many times.

3.6 Parallel Computation for Unbalanced Loads

The unbalanced loads can be computed using multi-processors at the element level. The stress for different element, and the unbalanced loads for each node for the elements are computed by different processors concurrently. Each processor will accumulate its own value of the unbalanced loads. Using Critical block, the contribution of each processor will be added. In the next statements, a brief algorithm shows the steps involved in parallel computation of the unbalanced loads:

```
      Presched do 10 I = 1 , Number_of_Elements
      Call subroutine stresses.
      Find the unbalance loads.
      Update the unbalanced load vector Δfp
10    CONTINUE
      Critical TYPE2
      do 30 I  = 1 , Number_of_Nodes
      Δf(I) = Δf(I) + Δfp(I)
20    CONTINUE
      End Critical
```

The vector $\{\Delta fp\}$ is declared as private real variable, and the vector $\{\Delta f\}$ is declared as shared real variable.

3.7 Structural Analysis Applications and Verifications

In this section the results obtained for N-R, mN-R, and BFGS methods are shown using the computer program NONDSA.FRC. The time used for generating and assembling the tangent stiffness matrix, calculate the unbalanced loads, total nonlinear analysis, total analysis time (include I/o) will be compared.

Example 1 Three-Dimensional Truss Structure

Two different sizes of trusses ,truss A and truss B, were used. The information needed to identify each truss is given in Figure 3.2. All the nodes are loaded in the X, Y and Z directions with forces according to a special pattern. The results for the first truss are given in Tables 3.10, 3.11, 3.12. The results for the second truss are given in Tables 3.13, 3.14, 3.15.

It is observed from Tables 3.10, and 3.13 that the speed up for solving the nonlinear problem using N-R method increases as the problem size increases. From Tables 3.10 and 3.13 one can calculate the speed up for the Choleski factorization for truss A to be (6.84) and for truss B (7.69) using eight processors. The lowest efficiency obtained for generating, and assembling the tangent stiffness matrix for truss A, and truss B is 95.84% using eight processors.

From Tables 3.13, 3.14, 3.15, the lowest total time obtained to complete nonlinear analysis for the two different trusses using only one processor is by the use of mN-R method. Using eight processors N-R method shows the least time for completing the nonlinear analysis part. N-R method is a better parallel-vector method than the mN-R method. It is also interesting to see that the number of

operation count for the N-R method is almost twice the number of operation count for the mN-R or the BFGS methods. The vector speed is an important factor in solving large problems. Choleski factorization and back substitution has a high vector speed compared to finding the unbalanced loads or the generation of the element stiffness matrices.

Example 2 Two-Dimensional Frame Structure

Two different sizes of two the dimensional frame,frame A and frame B, are used to provide the numerical performance of different methods. All the nodes are loaded in the X, Y and Z directions with forces, and also loaded about the X, Y and Z directions with moments according to special pattern. The information needed to identify the frame is given in Figure 3.3. The results for the first frame is given in Tables 3.16, 3.17, 3.18; the results for the second frame is given by tables 3.19, 3.20, 3.21.

The lowest efficiency obtained for generating and assembling the tangent stiffness matrix is 96.38% using six processors.

The least time obtained for the nonlinear analysis part is given by the use of mN-R method for frame A, using one or multi-processors. Using frame B where the number of D.O.F. increases, the use of N-R method for the nonlinear analysis shows the least time using single or multi-processors. By monitoring the speed up for factorizing the tangent stiffness matrix, using N-R method, for frame A and truss A, one can observe that the speed up for factorization, using eight processors, for frame A is (6.77), where the speed up for truss A is (7.60). The reason for that is the

bandwidth of frame A is (32) where the bandwidth of truss A is (172). The performance of the equation solver depends on the bandwidth, or the vector length.

By changing the number of loading steps from four steps to two steps for frame B, Table 3.22, mN-R method diverges where N-R and the BFGS methods converges.

Example 3 Three-Dimensional Control-Structure Interaction Model.

The required information for the control-structure interaction example Ref. [Belvin,1990] are given in Figure. 3.4. Only few of the nodes were loaded in the X, Y and Z directions with forces and also the same nodes were loaded with moments in about the X, Y and Z directions. The results for the above example are given in Tables 3.23,3.24,3.25.

The least efficiency obtained by the generating and assembling the tangent stiffness matrix is 89.40% by using eight processors.

3.8 Some Remarks On Nonlinear Structural Analysis

Based on the numerical results obtained from the above three structural examples, the following observations can be made:

1. In a sequential computer environment, the solution time for generating and assembly of the structural stiffness matrix is significantly smaller than the factorization time. However, the time ratio of (generation assembly and factorization) is relatively large in a parallel-vector computer environment (even for the case of one processor). This is due to the application of loop-unrolling technique and parallel processing equation solver [Agarwal at al.,1990].

2. The time required for generation and assembly of the structural stiffness matrix, matrix factorization, nonlinear analysis, entire nonlinear process is decreased as the number of processors is increased.

3. For linear static analysis, the solution time for the forward and backward elimination phase for solving a system of linear equations is insignificant as compared to the solution time for the factorization phase. For nonlinear static analysis, however, the forward and backward solution time can be a major component of the total nonlinear analysis time. This is especially true if the mN-R is employed.

4. Despite small half-bandwidth in all examples, the speed up factors for parallel factorization are still good. Speed up factors for the entire nonlinear analysis process (including all I/O as indicated in the last column of the Tables of the above examples, are still reasonably good.

5. In a sequential computer environment, the BFGS is usually considered to be the most effective method. In a parallel-vector computer environment, however, the N-R method seems to be the most efficient method.

6. In a parallel-vector computer environment, the BFGS and mN-R methods seems to be compatible (in terms of efficiency).

7. N-R method is numerically more reliable than m-N-R method in both sequential or parallel-vector computers.

8. The fewer number of operation count for any method does not mean that it requires less time than another method with a higher number of operation count in a parallel-vector computers. As seen from the results N-R method had a higher operation count, but the execution time was less than mN-R and the BFGS method.

9. The geometrically nonlinear analysis of the proposed procedure has been verified by comparing with the known solution in Ref. [Ryu et al.,1985]. Detail discussion of the numerical data in Ref. [Ryu et al.,1985] and the numerical comparison are given in Appendix B.

Table 3.1 Element Connectivity for Three Bar Truss.

Element Number	Node i	Node j
1	2	3
2	1	3
3	2	1

Table 3.2 Node Connectivity for Three Bar Truss.

Node Number	Global D.O.F	Elements with "node Number"	
		Node i	Node j
1	1,2	2	3
2	3,4	1,3	None
3	5,6	None	1,2

Table 3.3 Parallel Generation-Assembly of $K_{ii}^{(e)}$ for Three Bar Truss.

D.O.F	1	2	3	4	5	6
1	2	2				
2		2				
3			1,3	1,3		
4				1,3		
5						
6						

Table 3.4 Parallel Generation-Assembly of $K_{jj}^{(e)}$ for Three Bar Truss.

D.O.F.	1	2	3	4	5	6
1	3	3				
2		3				
3						
4						
5					1,2	1,2
6						1,2

Table 3.5 Parallel Generation-Assembly of $K_{ij}^{(e)}$ for Three Bar Truss.

D.O.F	1	2	3	4	5	6
1			3	3	2	2
2			3	3	2	2
3					1	1
4					1	1
5						
6						

Table 3.6 Linear Piecewise Time for Hybrid Choleski and P.C.G. Approach using one processor. (SECOND(),Reynolds).

Load Step No:	Choleski Method Time	Hybrid Approach Time	Number of Iterations
1	0.3418	0.3145	---
2	0.3181	0.3139	---
3	0.3576	0.0999	4
4	0.3338	0.1361	5
5	0.3351	0.1352	5
6	0.3582	0.1487	6
7	0.3279	0.1714	6
8	0.3651	0.1789	6
9	0.3393	0.1458	6
10	0.3215	0.1878	7
11	0.3348	0.1912	7
12	0.3251	0.1670	7
13	0.3655	0.1880	8
14	0.3399	0.1871	8
15	0.3527	0.1870	8
16	0.3167	0.2101	9
17	0.3582	0.2110	9
18	0.3505	0.2101	9
19	0.3251	0.2088	9
20	0.3185	0.2308	10

Total time for nonlinear analysis using Choleski = 11.6159 sec.
Total time for nonlinear analysis using Hybrid approach = 8.3368 sec.
Underline Time: indicate the use of Choleski in the hybrid approach.
Choleski Method Time includes the factorization time only.

Table 3.7 Linear Piecewise Time for Hybrid Choleski and P.C.G. Approach Using Four Processors. (SECOND(),Reynolds).

Load Step No:	Choleski Method Time	Hybrid Approach Time	Number of Iterations
1	0.0887	0.0824	---
2	0.0851	0.0823	---
3	0.0887	0.0611	4
4	0.0828	0.0761	5
5	0.0871	0.0761	5
6	0.0838	0.0893	6
7	0.0871	0.0825	---
8	0.0859	0.0615	6
9	0.0845	0.0765	6
10	0.0895	0.0771	7
11	0.0842	0.0760	7
12	0.0914	0.0904	7
13	0.0833	0.0814	---
14	0.0821	0.0312	4
15	0.0830	0.0765	5
16	0.0938	0.0759	5
17	0.0832	0.0767	5
18	0.0868	0.0902	6
19	0.0874	0.0816	---
20	0.0914	0.0615	4

Total time for nonlinear analysis using Choleski = 3.1252 sec.
Total time for nonlinear analysis using Hybrid approach = 2.7246 sec.
Underline Time: indicate the use of Choleski in the hybrid approach.
Choleski Method Time includes the factorization time only.

Table 3.8 Linear Piecewise Time for Hybrid Choleski and P.C.G. Approach Using Eight Processors. (SECOND(),Reynolds).

Load Step No:	Choleski Method Time	Hybrid Approach Time	Number of Iterations
1	0.0411	0.0417	---
2	0.0449	0.0449	---
3	0.0423	0.0547	4
4	0.0436	0.0410	---
5	0.0413	0.0535	4
6	0.0419	0.0416	---
7	0.0453	0.0557	4
8	0.0484	0.0417	---
9	0.0416	0.0547	4
10	0.0455	0.0408	---
11	0.0418	0.0547	4
12	0.0433	0.0416	---
13	0.0418	0.0548	4
14	0.0446	0.0409	---
15	0.0434	0.0554	4
16	0.0434	0.0416	---
17	0.0418	0.0551	4
18	0.0415	0.0416	---
19	0.0451	0.0544	4
20	0.0454	0.0409	---

Total time for nonlinear analysis using Choleski = 1.7379 sec.
Total time for nonlinear analysis using Hybrid approach = 1.7248 sec.
Underline Time: indicate the use of Choleski in the hybrid approach.
Choleski Method Time includes the factorization time only.

Table 3.9 S.O.R. method for three dimensional truss (1872 D.O.F)

	Number of processors				
	1	2	4	6	8
NTI	172	182	183	186	231
Time	3.6372	1.8556	1.0183	0.6744	0.6521
Speed Up	-----	1.9601	3.5718	5.3932	5.5777
Efficiency	100.0000	98.0100	89.3000	89.8900	69.7200

Table 3.10 N-R Method for Three Dimensional Truss (1872 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	8	8	8	8	8
T.GAS	1.4698	0.7574	0.3819	0.2524	0.1917
T.FAC	2.5267	1.3109	0.6571	0.4401	0.3313
T.BAC	0.0976	0.0968	0.0970	0.0971	0.0972
T.UFO	0.1424	0.7141	0.0360	0.0242	0.1845
T.NON	4.4103	2.3261	1.1827	0.8247	0.6619
T.TOT	4.9351	2.7000	1.7057	1.3470	1.1843
S.GAS	----	1.9406	3.8487	5.8233	7.6672
S.NON	----	1.8960	3.7290	5.3478	6.6631
S.TOT	----	1.8278	2.8933	3.6638	4.1671
E.GAS	100.0000	97.0300	96.2200	97.0600	95.8400
E.NON	100.0000	94.8000	93.2300	89.1300	83.2900
E.TOT	100.0000	91.3900	72.3300	61.0600	52.0900

Total number of operation for the nonlinear analysis = 338 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.11 mN-R Method for Three Dimensional Truss (1872 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	23	23	23	23	23
T.GAS	0.7321	0.3781	0.1909	0.1261	0.0955
T.FAC	1.2583	0.6559	0.3274	0.2187	0.1648
T.BAC	0.2774	0.2789	0.2779	0.2786	0.2773
T.UFO	0.3915	0.1965	0.0990	0.0665	0.0506
T.NON	3.1347	1.6044	1.0083	0.7726	0.6580
T.TOT	3.6560	2.2185	1.5308	1.2947	1.1793
S.GAS	-----	1.9363	3.8350	5.8057	7.6660
S.NON	-----	1.9538	3.1089	4.0573	4.7640
S.TOT	-----	1.6480	2.3883	2.8238	3.1001
E.GAS	100.0000	96.8200	95.8800	96.7600	95.8300
E.NON	100.0000	97.6900	77.7200	67.6200	59.5500
E.TOT	100.0000	82.4000	59.7100	47.0600	38.7500

Total number of operation for the nonlinear analysis = 219 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.12 BFGS Method for Three Dimensional Truss (1872 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	17	17	17	17	17
T.GAS	0.7253	0.3800	0.1890	0.1261	0.0982
T.FAC	1.2636	0.6778	0.3294	0.2230	0.1746
T.BAC	0.0497	0.0498	0.0542	0.0478	0.0548
T.UFO	0.1463	0.0754	0.0387	0.0270	0.0211
T.NON	4.6108	2.3570	1.3510	1.0273	0.8404
T.TOT	5.1362	2.9653	1.8728	1.3204	1.3646
S.GAS	-----	1.9087	3.8376	5.7518	7.3859
S.NON	-----	1.9562	3.4129	4.4883	5.4864
S.TOT	-----	1.7321	2.7425	3.8899	3.7639
E.GAS	100.0000	95.4400	95.9400	95.8600	92.3200
E.NON	100.0000	97.8100	85.3200	74.8100	68.5800
E.TOT	100.0000	86.6100	68.5600	64.8300	47.0500

Total number of operation for the nonlinear analysis = 219 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.13 N-R Method for Three Dimensional Truss (2790 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	10	10	10	10	10
T.GAS	2.7574	1.4263	0.7131	0.4752	0.3584
T.FAC	6.1057	3.1545	1.5851	1.0553	0.7935
T.BAC	0.2006	0.2024	0.2022	0.2008	0.2019
T.UFO	0.2684	0.1346	0.0678	0.4554	0.0344
T.NON	9.6604	5.0854	2.6550	1.8355	1.4357
T.TOT	10.4443	5.5698	3.4381	2.3718	2.2183
S.GAS	-----	1.9333	3.8668	5.8026	7.6936
S.NON	-----	1.8996	3.6386	5.2631	6.7287
S.TOT	-----	1.8752	3.0378	4.4035	4.7082
E.GAS	100.0000	96.6700	96.6700	96.7100	96.1700
E.NON	100.0000	94.9800	90.9700	87.7200	84.1100
E.TOT	100.0000	94.9800	90.9700	73.3900	58.8500

Total number of operation for the nonlinear analysis = 1023 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.14 mN-R Method for Three Dimensional Truss (2790 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	57	57	57	57	57
T.GAS	1.1021	0.5700	0.2849	0.1896	0.1431
T.FAC	2.4429	1.2611	0.6333	0.4204	0.3184
T.BAC	1.1443	1.1289	1.1481	1.1425	1.5101
T.UFO	1.2632	0.6439	0.3308	0.2284	0.1774
T.NON	7.9697	4.5993	2.9203	2.3480	2.0900
T.TOT	8.7507	5.3822	3.7037	3.1225	2.8755
S.GAS	----	1.9335	3.8684	5.8128	7.7016
S.NON	----	1.7328	2.7291	3.3943	3.8133
S.TOT	----	1.6259	2.3627	2.8025	3.0432
E.GAS	100.0000	96.6800	96.7100	96.8800	96.2700
E.NON	100.0000	86.6400	68.2300	56.5700	47.6700
E.TOT	100.0000	81.3000	59.0700	46.7100	38.0400

Total number of operation for the nonlinear analysis = 553 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.15 BFGS Method for Three Dimensional Truss (2790 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	16	16	16	16	16
T.GAS	1.6404	0.8581	0.4216	0.2819	0.2142
T.FAC	3.6867	1.9702	1.0077	0.6507	0.4968
T.BAC	0.1214	0.1278	0.1282	0.1288	0.1323
T.UFO	0.3278	0.1671	0.0854	0.0596	0.0457
T.NON	10.1073	5.2596	3.0443	2.1723	1.7793
T.TOT	10.9033	6.0918	3.3162	2.9545	2.5952
S.GAS	----	1.9117	3.8909	5.8191	7.6583
S.NON	----	1.9217	3.3201	4.6528	5.6805
S.TOT	----	1.7898	3.2879	3.6904	4.2013
E.GAS	100.0000	95.5900	97.2700	96.9900	95.7300
E.NON	100.0000	96.0900	83.0000	77.5500	71.0100
E.TOT	100.0000	89.4900	82.2000	61.5100	52.5200

Total number of operation for the nonlinear analysis = 665 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.16 N-R Method for Two Dimensional Frame (880 D.O.F),

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
NTI	27	27	27	27	27
T.GAS	3.9052	1.9602	0.9780	0.6592	0.4899
T.FAC	0.8838	0.5286	0.26928	0.1758	0.1338
T.BAC	0.1122	0.1132	0.1132	0.1130	0.1130
T.UFO	3.4979	1.7516	0.8765	0.5846	0.4405
T.NON	15.4178	7.8722	3.9882	2.7083	2.0632
T.TOT	15.7652	8.1127	4.3360	3.0570	2.4121
S.GAS	----	1.9922	3.9930	5.9242	7.9714
S.NON	----	1.9585	3.8659	5.6928	7.4728
S.TOT	----	1.9433	3.6359	5.1571	6.5359
E.GAS	100.0000	99.6100	99.8300	98.7400	99.6400
E.NON	100.0000	97.9300	96.6500	94.8800	93.4100
E.TOT	100.0000	97.1700	90.9000	85.9500	81.7000

Total number of operation for the nonlinear analysis = 430 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.17 mN-R Method for Two Dimensional Frame (880 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
NTI	30	30	30	30	30
T.GAS	1.1665	0.5829	0.2914	0.1967	0.1462
T.FAC	0.2587	0.1539	0.0726	0.0529	0.0393
T.BAC	0.1245	0.1245	0.1249	0.1209	0.1251
T.UFO	4.2442	2.1222	1.0629	0.7105	0.5534
T.NON	13.1589	6.6248	3.4058	2.3141	1.7781
T.TOT	13.5050	6.9710	3.7519	2.5417	2.1239
S.GAS	----	2.0012	4.0031	5.9304	7.9788
S.NON	----	1.9863	3.8637	5.6864	7.4005
S.TOT	----	1.9373	3.5995	5.3134	6.3586
E.GAS	100.0000	100.0600	100.0800	98.8400	99.7400
E.NON	100.0000	99.3200	96.5900	94.7700	92.5100
E.TOT	100.0000	96.8700	89.9900	88.5600	79.4800

Total number of operation for the nonlinear analysis = 360 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.18 BFGS Method for Two Dimensional Frame (880 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
NTI	22	22	22	22	22
T.GAS	2.7913	1.4058	0.7041	0.4747	0.3527
T.FAC	0.6121	0.3682	0.1854	0.1248	0.0938
T.BAC	0.07882	0.0794	0.0796	0.7961	0.0795
T.UFO	7.3999	3.6975	1.8606	1.2422	0.9385
T.NON	18.3471	9.1888	4.7393	3.2122	2.4508
T.TOT	18.6964	9.6386	5.0872	3.5602	2.7975
S.GAS	----	1.9856	3.9644	5.8801	7.9141
S.NON	----	1.9967	3.8713	5.7117	7.4862
S.TOT	----	1.9397	3.6752	5.2515	6.6833
E.GAS	100.0000	99.2800	99.1100	98.0000	98.9300
E.NON	100.0000	99.8400	96.7800	95.2000	93.5800
E.TOT	100.0000	96.9900	91.8800	87.5300	83.5400

Total number of operation for the nonlinear analysis = 325 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.19 N-R Method for Two Dimensional Frame (2520 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
NTI	38	38	38	38	38
T.GAS	16.3223	8.1669	4.0847	2.8225	2.0472
T.FAC	4.6167	2.6371	1.3373	0.8862	0.6796
T.BAC	0.4569	0.4591	0.4581	0.4576	0.4580
T.UFO	19.5624	9.7921	4.9069	3.2750	2.4553
T.NON	65.2746	32.7539	16.8468	11.3502	8.7007
T.TOT	66.2855	33.7726	17.8664	12.3632	9.7161
S.GAS	----	1.9986	3.9960	5.7829	7.9730
S.NON	----	1.9929	3.8746	5.7510	7.5022
S.TOT	----	1.9627	3.7101	5.3615	6.8222
E.GAS	100.0000	99.9300	99.9000	96.3800	99.6600
E.NON	100.0000	99.6500	96.8700	95.8500	93.7800
E.TOT	100.0000	98.1400	92.7500	89.3600	85.2800

Total number of operation for the nonlinear analysis = 850 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.20 mN-R Method for Two Dimensional Frame (2520 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
NTI	54	54	54	54	54
T.GAS	3.4510	1.7158	0.8605	0.5939	0.4307
T.FAC	0.9745	0.5548	0.2757	0.1866	0.1465
T.BAC	0.6519	0.2796	0.6526	0.6537	0.6533
T.UFO	33.3980	16.7012	8.3636	5.5850	4.1911
T.NON	67.4899	33.7313	17.4189	11.8355	9.0715
T.TOT	68.5020	34.8288	18.4322	12.8476	10.0845
S.GAS	----	2.0113	4.0105	5.8107	8.0125
S.NON	----	2.0008	3.8745	5.7023	7.4398
S.TOT	----	1.9668	3.7164	5.3319	6.7928
E.GAS	100.0000	100.5700	100.2600	96.8500	100.1600
E.NON	100.0000	100.0400	96.8600	95.0400	93.0000
E.TOT	100.0000	98.3400	92.9100	88.8700	84.9100

Total number of operation for the nonlinear analysis = 694 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.21 BFGS Method for Two Dimensional Frame (2520 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	43	43	43	43	43
T.GAS	16.3502	8.2046	4.1082	2.8362	2.1309
T.FAC	4.5549	2.6438	1.2655	0.9112	0.6438
T.BAC	0.4539	0.4561	0.4549	0.4547	0.4548
T.UFO	43.3875	21.7438	10.8973	7.2855	6.1083
T.NON	83.2732	42.2580	21.4756	13.9066	11.6587
T.TOT	84.2856	43.2735	22.4870	15.5081	11.7327
S.GAS	----	1.9928	3.9799	5.7648	7.6729
S.NON	----	1.9706	3.8776	5.9880	7.1426
S.TOT	----	1.9477	3.7482	5.4349	7.1838
E.GAS	100.0000	99.6400	99.5000	96.0800	95.9100
E.NON	100.0000	98.5300	96.9400	99.8000	89.2800
E.TOT	100.0000	97.3900	93.7100	90.5800	89.8000

Total number of operation for the nonlinear analysis = 725 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.22 Two Dimensional Frame (2520 D.O.F) with Only one Processor, and Number of Loading Steps is Two.

	Method		
	N-R	mN-R	BFGS
NLS	2	2	2
NTI	25	101	37
T.GAS	10.7513	1.7159	11.0210
T.FAC	3.0532	0.4878	3.1290
T.BAC	0.3006	1.3256	0.3111
T.UFO	20.7560	115.1973	29.7200
T.NON	43.0076	130.5540	52.5112
T.TOT	44.0184	131.5683	53.5223

Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.23 N-R Method for CSI Structure (Beam Element) (3096 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
NTI	25	25	25	25	25
T.GAS	5.3604	2.7537	1.4716	0.9320	0.7494
T.FAC	4.6840	2.6067	1.3212	0.8930	0.6722
T.BAC	0.3996	0.4000	0.3998	0.4000	0.3997
T.UFO	2.7328	1.3705	0.6878	0.4623	0.3468
T.NON	20.2866	10.6845	5.3859	3.8852	2.9208
T.TOT	20.7095	10.6846	5.8075	3.8855	2.9211
S.GAS	----	1.9466	3.6426	5.7515	7.1529
S.NON	----	1.8987	3.7666	5.2215	6.9456
S.TOT	----	1.9383	3.5660	5.3299	7.0896
E.GAS	100.0000	97.3300	91.0700	95.8600	89.4100
E.NON	100.0000	94.9400	94.1700	87.0300	86.8200
E.TOT	100.0000	96.9200	89.1500	88.8300	88.6200

Total number of operation for the nonlinear analysis = 438 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.24 mN-R Method for CSI Structure (Beam Element) (3096 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
NTI	47	47	47	47	47
T.GAS	1.7153	0.8826	0.4711	0.2981	0.2403
T.FAC	1.4977	0.8414	0.4225	0.2850	0.2142
T.BAC	0.9100	0.9119	0.9113	0.9116	0.9127
T.UFO	7.4188	3.7131	1.8665	1.2514	0.9465
T.NON	26.4865	13.4139	7.3532	5.2554	4.1977
T.TOT	26.9086	13.8369	7.7759	5.6783	4.1981
S.GAS	----	1.9435	3.6411	5.7541	7.1382
S.NON	----	1.9746	3.6020	5.0399	6.3098
S.TOT	----	1.9447	3.4605	4.7388	6.4097
E.GAS	100.0000	97.1800	91.0300	95.9000	89.2300
E.NON	100.0000	98.7300	90.0500	84.0000	78.8700
E.TOT	100.0000	97.2400	86.5100	78.9800	80.1200

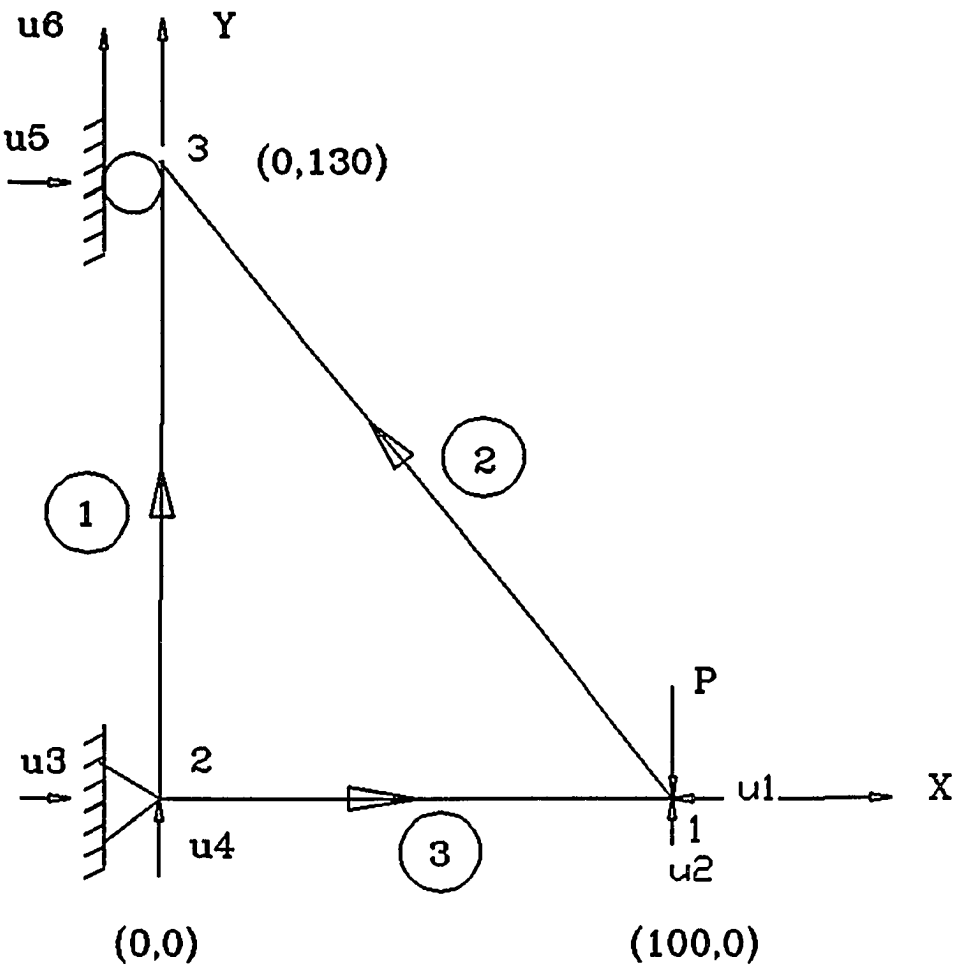
Total number of operation for the nonlinear analysis = 301 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 3.25 BFGS Method for CSI Structure (Beam Element) (3096 D.O.F).

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
NTI	26	26	26	26	26
T.GAS	4.727	2.4190	1.2924	0.8191	0.6607
T.FAC	4.139	2.3122	1.1740	0.7879	0.5868
T.BAC	0.3515	0.3457	0.3488	0.3516	0.3514
T.UFO	8.6107	4.3505	2.1684	1.4582	1.1002
T.NON	23.4825	12.0501	6.2200	4.3928	3.3951
T.TOT	23.9106	12.4718	6.6398	4.8136	3.8168
S.GAS	----	1.9541	3.6575	5.7710	7.1545
S.NON	----	1.9487	3.7753	5.3457	6.9166
S.TOT	----	1.9172	3.6011	4.9673	6.2646
E.GAS	100.0000	97.7100	91.4400	96.1800	89.4300
E.NON	100.0000	97.4400	94.3800	89.1000	86.4600
E.TOT	100.0000	95.8600	90.0300	82.7900	78.3100

Total number of operation for the nonlinear analysis = 354 M. operations.
Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Figure 3.1 Three Bar Truss.

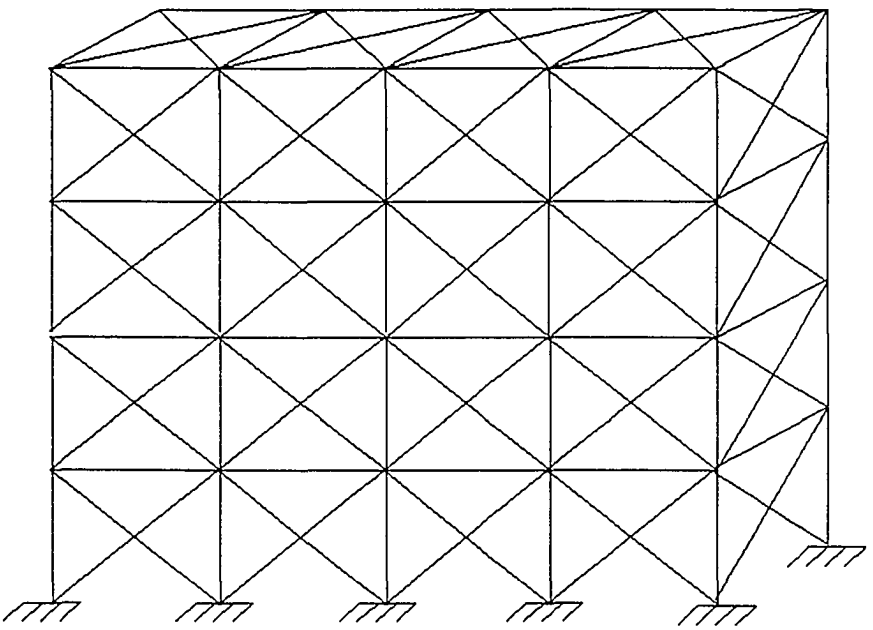


P = 1000.0 Kips

A_1 = 1.20 in^2 A_2 = 2.00 in^2 A_3 = 3.00 in^2

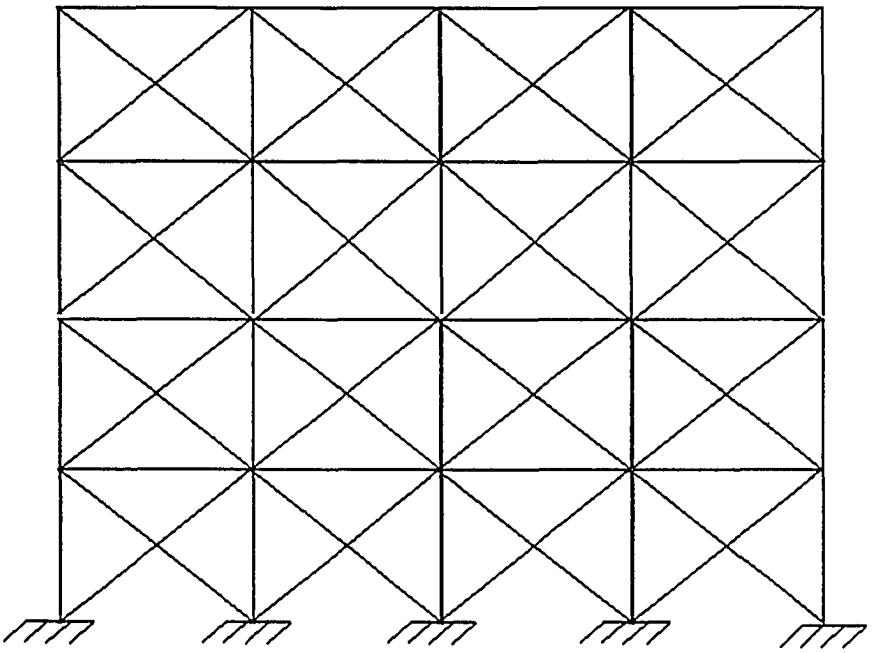
E = 29000.00 $Kips/in^2$

Figure 3.2 Typical Three Dimensional Truss.



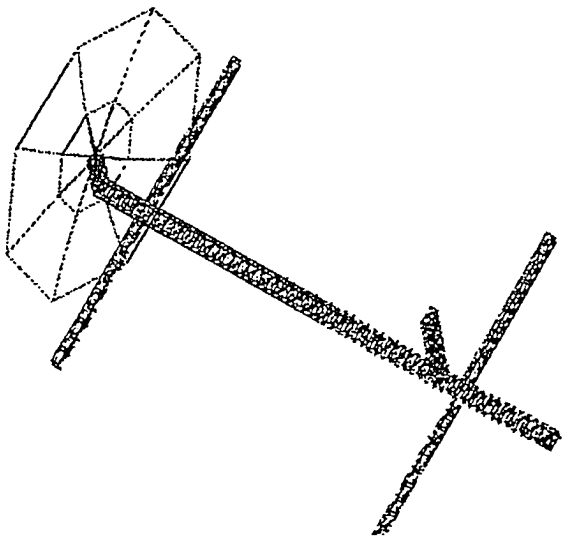
	Truss A	Truss B
Number of Stories	12	15
Number of Bays	25	30
Number of D.O.F	1872	2790
Maximum bandwidth	172	202
Number of elements	4012	5986
Number of nodes	676	922

Figure 3.3 Typical Two Dimensional Frame.



	Frame A	Frame B
Number of Stories	41	61
Number of Bays	10	20
Number of D.O.F	880	2520
Maximum bandwidth	32	52
Number of elements	1640	4860
Number of nodes	452	1282

Figure 3.4 CSI Structure Model.



Number of D.O.F	3096
Maximum bandwidth	108
Number of elements	1647
Number of nodes	537

Figure 3.5 Time for Nonlinear Finite Element Analysis.
Three Dimensional Truss (1872 D.O.F).

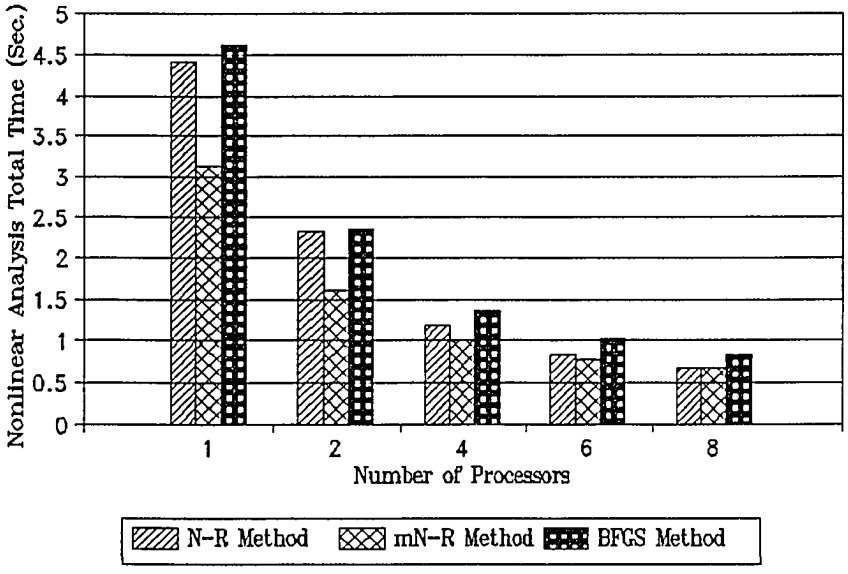


Figure 3.6 Speed Up for Nonlinear Finite Element Analysis. Three Dimensional Truss (1872 D.O.F).

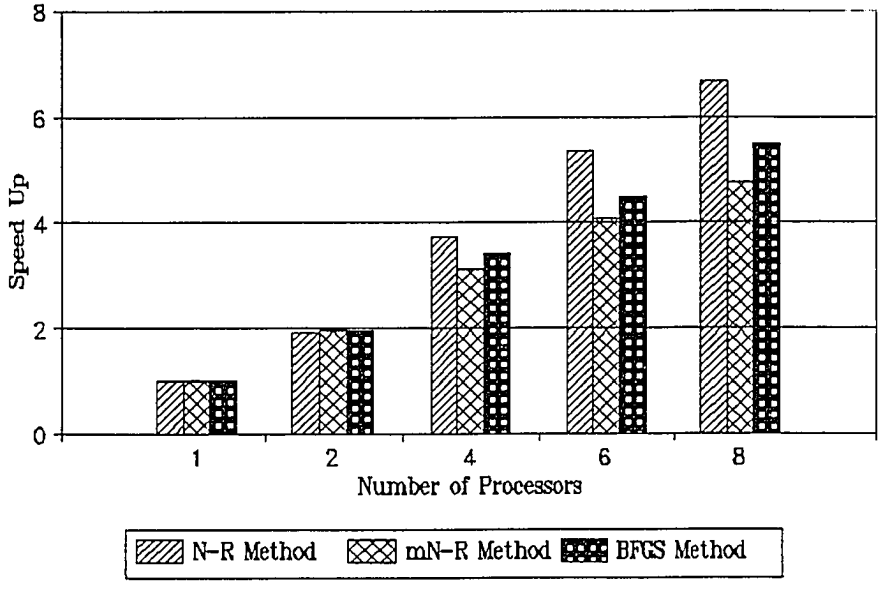


Figure 3.7 Efficiency for Nonlinear Finite Element Analysis. Three Dimensional Truss (1872 D.O.F).

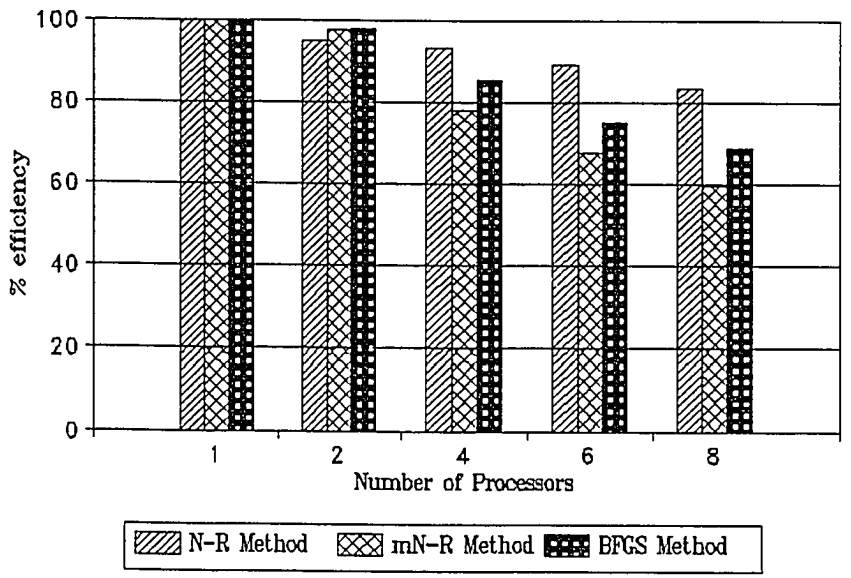


Figure 3.8 Time for Nonlinear Finite Element Analysis.
Three Dimensional Truss (2790 D.O.F).

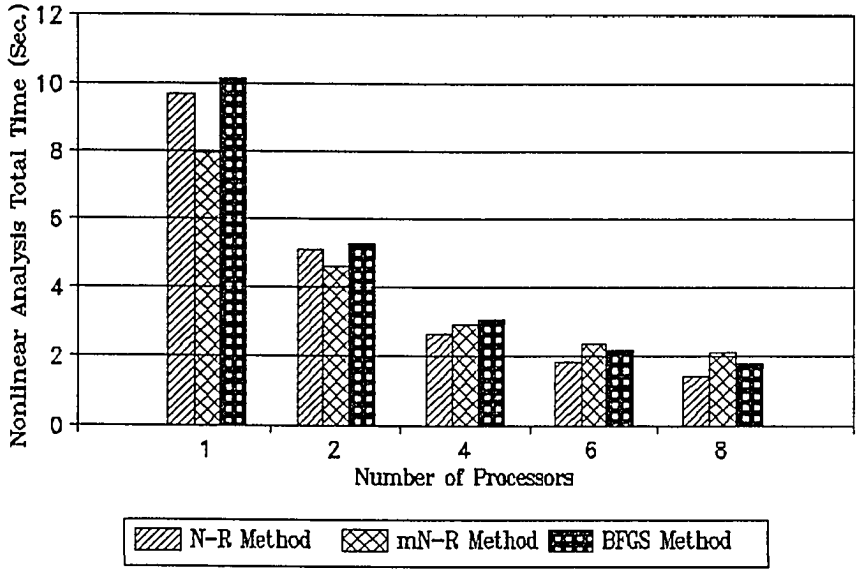


Figure 3.9 Speed Up for Nonlinear Finite Element Analysis. Three Dimensional Truss (2790 D.O.F).

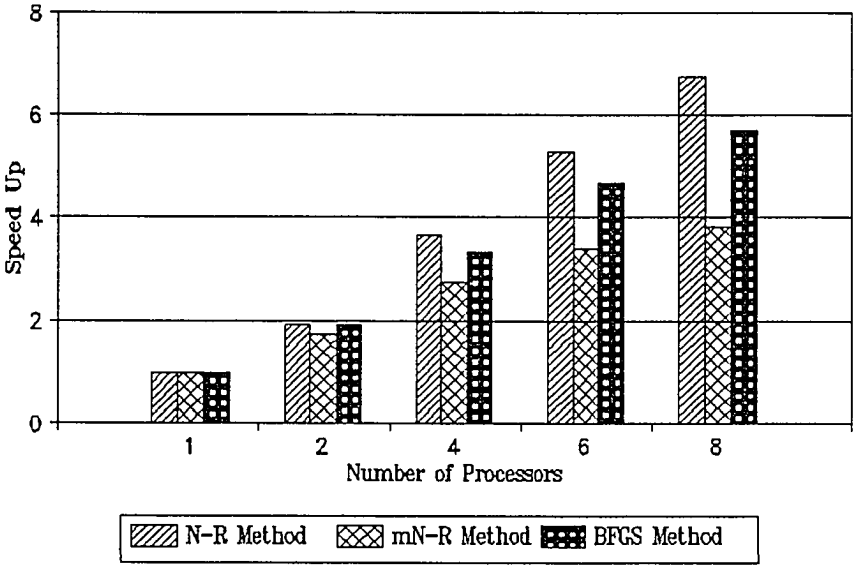


Figure 3.10 Efficiency for Nonlinear Finite Element Analysis. Three Dimensional Truss (2790 D.O.F).

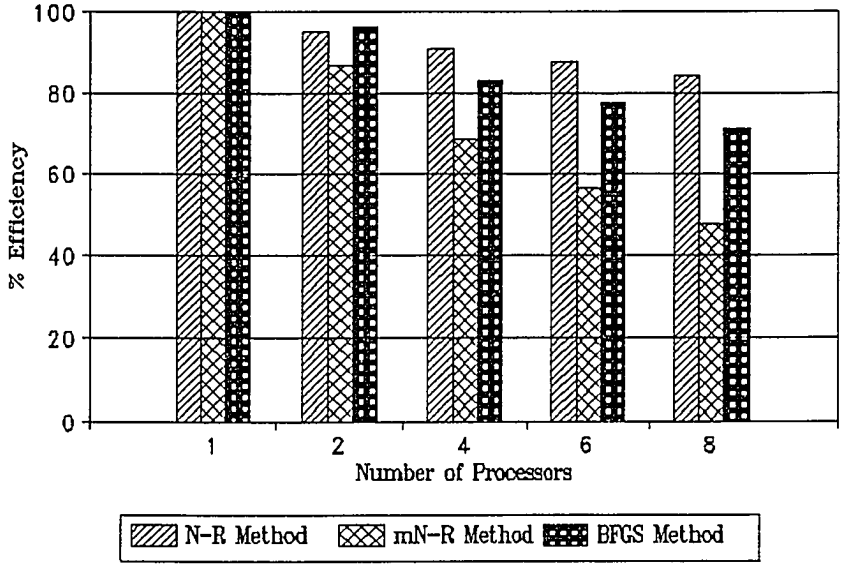


Figure 3.11 Time for Nonlinear Finite Element Analysis.
Two Dimensional Frame (880 D.O.F).

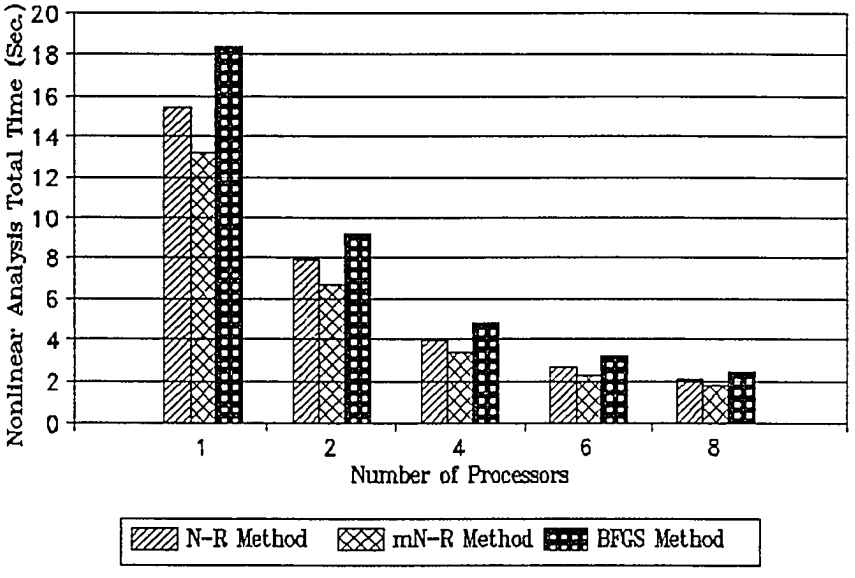


Figure 3.12 Speed Up for Nonlinear Finite Element Analysis. Two Dimensional Frame (880 D.O.F).

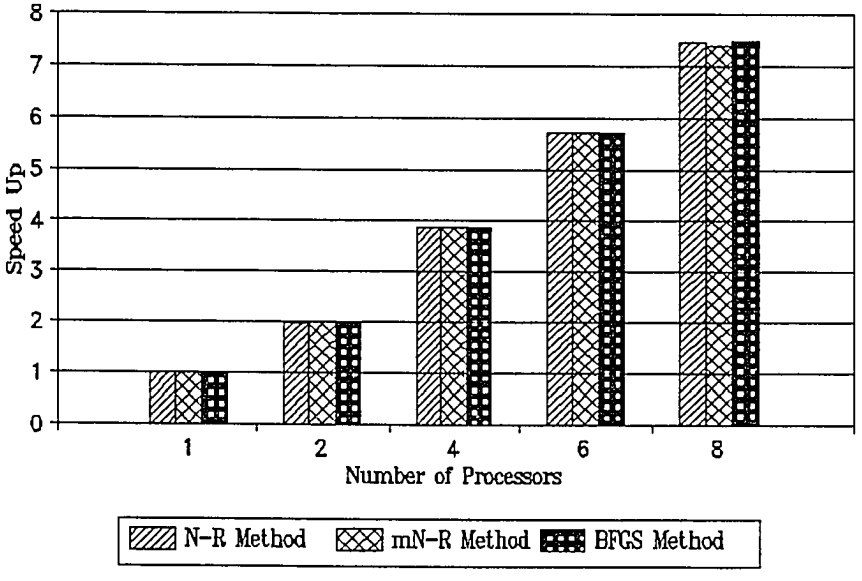


Figure 3.13 Efficiency for Nonlinear Finite Element Analysis. Two Dimensional Frame (880 D.O.F).

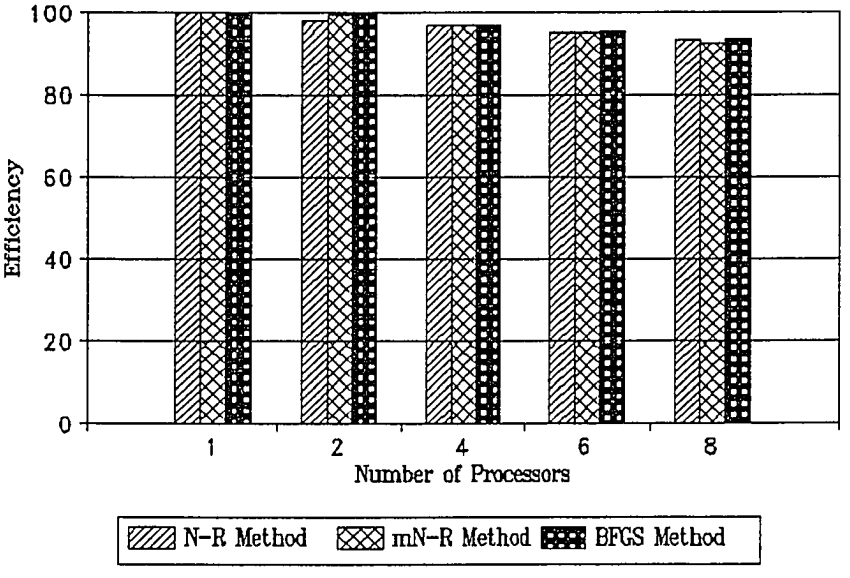


Figure 3.14 Time for Nonlinear Finite Element Analysis.
Two Dimensional Frame (2520 D.O.F).

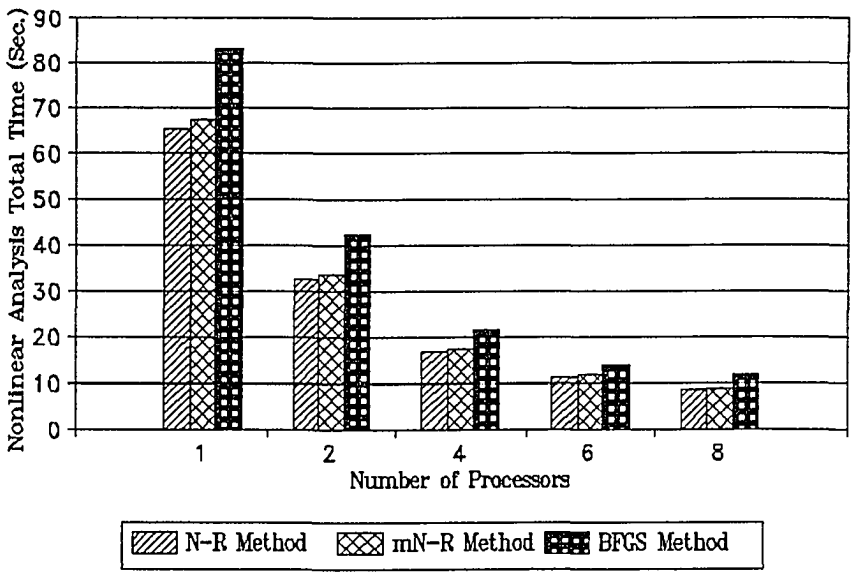


Figure 3.15 Speed Up for Nonlinear Finite Element Analysis. Two Dimensional Frame (2520 D.O.F).

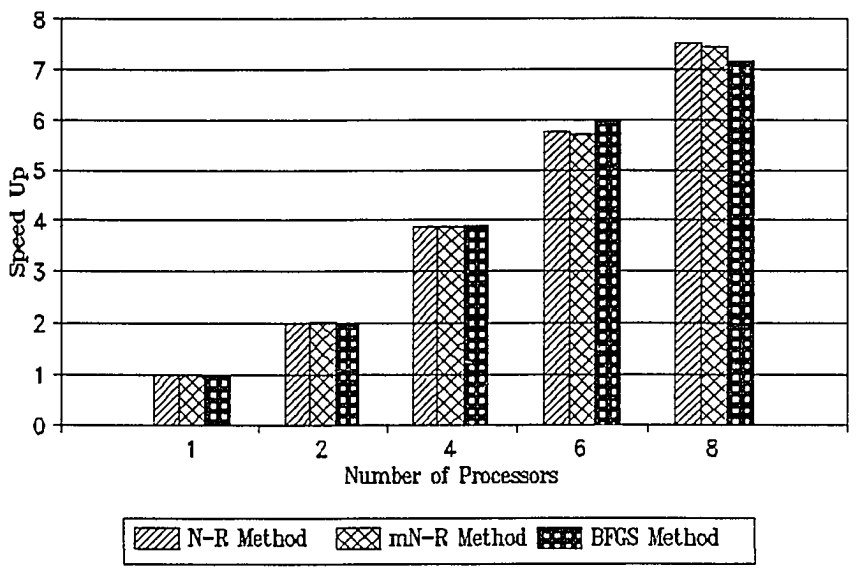


Figure 3.16 Efficiency for Nonlinear Finite Element Analysis. Two Dimensional Frame (2520 D.O.F).

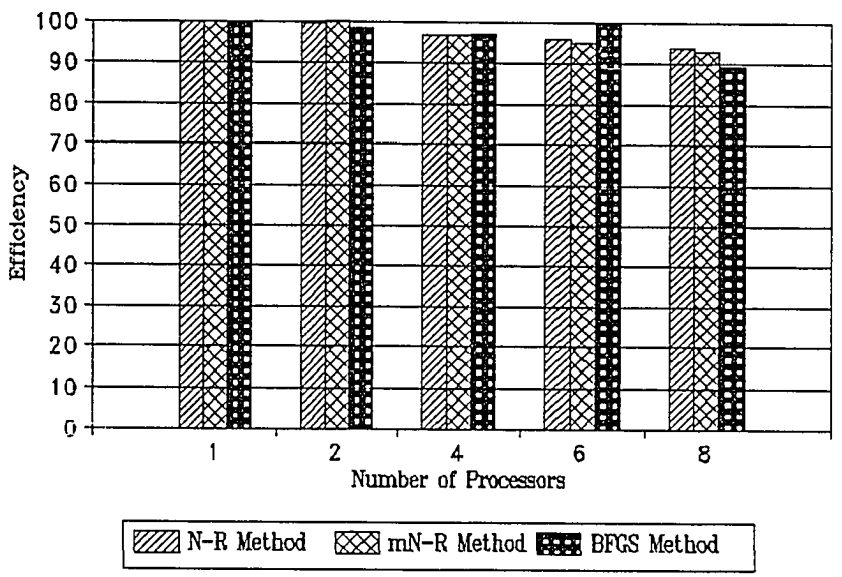


Figure 3.17 Time for Nonlinear Finite Element Analysis.
CSI Structure (3096 D.O.F).

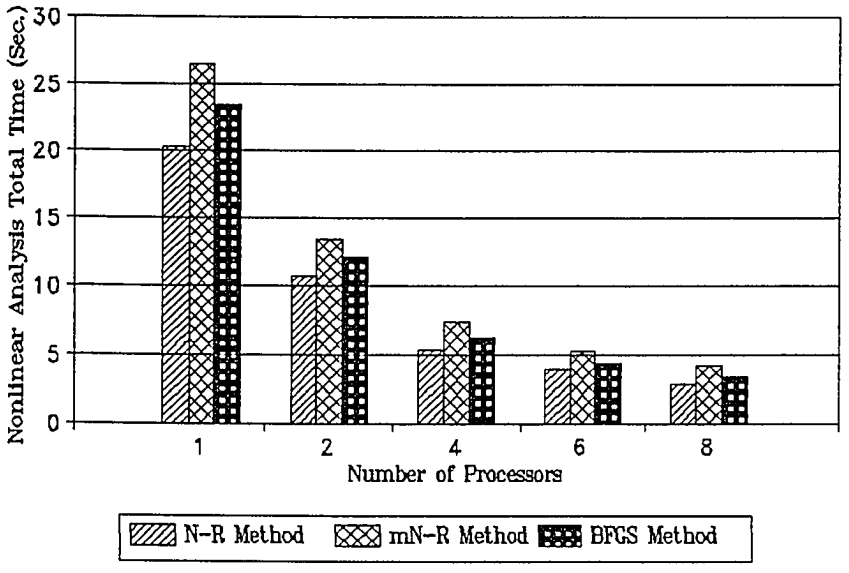


Figure 3.18 Speed Up for Nonlinear Finite Element Analysis. CSI Structure (3096 D.O.F).

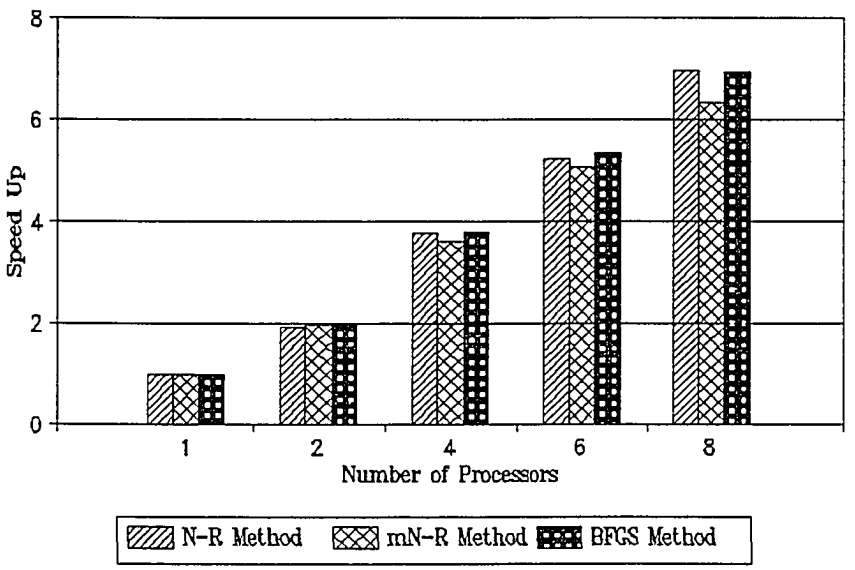
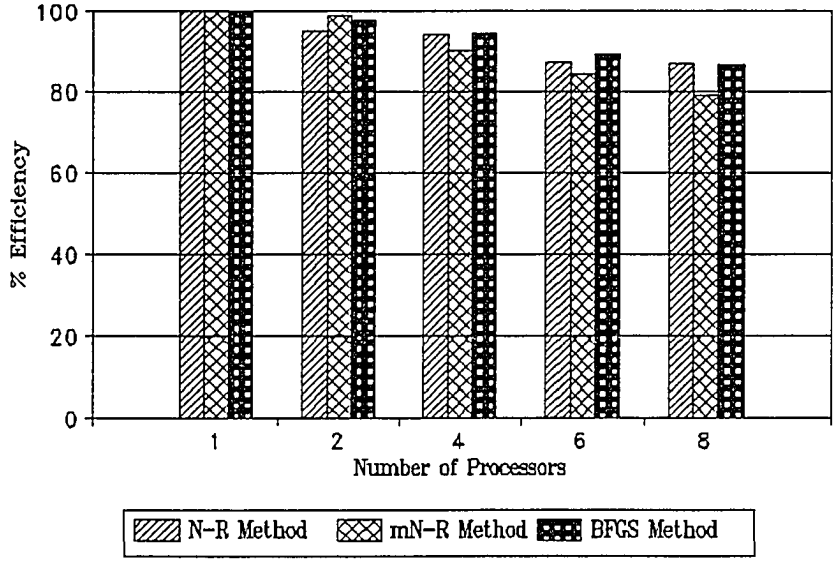


Figure 3.19 Efficiency for Nonlinear Finite Element Analysis. CSI Structure (3096 D.O.F).



CHAPTER 4

PARALLEL-VECTOR COMPUTATION FOR GEOMETRICALLY NONLINEAR DESIGN SENSITIVITY ANALYSIS

4.1 General Procedures For Nonlinear Design Sensitivity Analysis

In optimization, one needs to minimize an objective function subjected to a given set of constraints, such as

$$\text{minimize} \quad \psi_o(b,U) \tag{4.1}$$

$$\text{subjected to} \quad \psi_i(b,U) \leq 0 \qquad i = 1,NC \tag{4.2}$$

where b is the design variable vector, U is the displacement vector, and NC means the number of constraints. For structural analysis applications, one needs to minimize the weight of the structure subjected to the displacement and stress constraints. To solve the optimization problem, we need to know the derivatives of the objective and constraint functions with respect to the design variable:

$$\frac{d\psi}{db} = \frac{\partial\psi}{\partial b} + \left(\frac{\partial\psi}{\partial U}\right)\frac{dU}{db} \tag{4.3}$$

where the terms $\partial\psi/\partial U$ and $\partial\psi/\partial b$ are easy to calculate since the constraint function ψ is explicitly written in terms of U and b. To find the (dU/db), one uses the equilibrium equation:

$$Q(b,U) = R - F = 0 \tag{4.4}$$

where F is the externally applied loads and R is the internally resisting loads. Taking total derivatives with respect to the design variable b, the following equations are obtained:

$$\frac{\partial Q}{\partial b} + \left(\frac{\partial Q}{\partial U}\right) \frac{dU}{db} = 0 \tag{4.5}$$

where

$$\frac{\partial Q}{\partial U} = \frac{\partial R}{\partial U} - \frac{\partial F}{\partial U} \tag{4.6}$$

$$\frac{\partial Q}{\partial b} = \frac{\partial R}{\partial b} - \frac{\partial F}{\partial b} \tag{4.7}$$

where $(\partial Q/\partial U)$ is the tangent stiffness matrix $[K_T]$. In the present study, it is assumed that the external loads are independent of the displacement vector U and the design variable b, then $(\partial F/\partial U)$ and $(\partial F/\partial b)$ are equal to zero. Thus equation (4.5) can be simplified as:

$$[K_T] \frac{dU}{db} = - \frac{\partial R}{\partial b} \tag{4.8}$$

The above derivation is called the direct method. From equation (4.8), one can derive the adjoint variable method by pre-multiplying equation (4.8) by λ^T:

$$(\lambda^T * [K_T]) \left(\frac{dU}{db}\right) = - \lambda^T * \left(\frac{\partial R}{\partial b}\right) \tag{4.9}$$

such that λ will satisfy the adjoint equation:

$$[K_T] * \lambda = \left(\frac{\partial \psi}{\partial U}\right)^T \tag{4.10}$$

then equation (4.9) can be written as:

$$\frac{\partial \psi}{\partial U}\left(\frac{dU}{db}\right) = - \lambda^T \left(\frac{\partial R}{\partial b}\right) \tag{4.11}$$

using either equation (4.8), or (4.11) in equation (4.3) will yield the same results.

In the above analysis, the areas of the cross section are used as the design variables for the truss structure. For the frame structure, the hollow circular section is chosen, where the outer diameter is chosen as the design variable.

4.1.1 Displacement Constraints

The displacement constraint equation for the truss and the frame structure is given by:

$$\psi_i = \frac{u_i}{u_j^a} - 1.0 \leq 0.0 \qquad i = 1 , m \tag{4.12}$$

where m is the number of displacement constraints, u_j^a is the allowable displacement at the j^{th} degree of freedom.

$$\frac{\partial \psi_i}{\partial u_j} = \frac{1.0}{u_j^a} \qquad \text{for } i = j \tag{4.13}$$

$$\frac{\partial \psi_i}{\partial u_j} = 0.0 \qquad \text{for } i \neq j \tag{4.14}$$

using equations (4.10) and (4.11) one can obtain (dU/db). This will complete the calculations for the design sensitivity of the displacement constraints.

4.1.2 Stress Constraints

The stress constraint equation is given by:

$$\psi_i = \frac{\sigma_i}{\sigma_j^a} - 1.0 \leq 0.0 \qquad i = 1 , k \tag{4.15}$$

where k is the number of stress constraints equations, σ_j^a is the allowable stress for element j. For truss structures, the stress is given by:

$$\sigma_j = \frac{P_j}{A_j} = \frac{E}{L_j} * (u_1 - u_2) \tag{4.16}$$

P_j is the axial force in element j and A_j, L_j is the area and the length of element j, respectively. u_2, u_1 are the displacement of element j at nodes two and one.

$$\frac{\partial \psi_i}{\partial u_1} = \frac{1.0}{\sigma_j^a} * \left(\frac{E}{L_j}\right) \tag{4.17}$$

$$\frac{\partial \psi_i}{\partial u_2} = - \frac{1.0}{\sigma_j^a} * \left(\frac{E}{L_j}\right) \tag{4.18}$$

$$\frac{\partial \psi_i}{\partial b} = 0 \tag{4.19}$$

σ_j for two dimensional frame can be given as:

$$\sigma_j = \frac{P_j}{A_j} \pm \frac{M_{jz}\, c_j}{I_j} \tag{4.20}$$

$$\sigma_j = \frac{E}{L_j} * (u_1 - u_4) \pm \frac{E}{L_j^2} * (6*u_2 + 4*L_j*u_3 - 6*u_5 + 2*L_j*u_6) * c_j \tag{4.21}$$

where M_{jz} is the maximum moment about the z direction for element j, I_j, L_j, c_j is the moment of inertia, the length of element j, and the distance from the center of the section to the extreme fiber. u_1 through u_6 is the local displacement vector for element j.

$$\frac{\partial \psi_i}{\partial u_1} = \frac{1.0}{\sigma_j^a} * \left(\frac{E}{L_j}\right) \tag{4.22}$$

$$\frac{\partial \psi_i}{\partial u_2} = \frac{6.0}{\sigma_j^a} * \left(\frac{E}{L_j^2}\right) * c_j \tag{4.23}$$

$$\frac{\partial \psi_i}{\partial u_3} = \frac{4.0}{\sigma_j^a} * \left(\frac{E}{L_j}\right) * c_j \tag{4.24}$$

$$\frac{\partial \psi_i}{\partial u_4} = \frac{-1.0}{\sigma_j^a} * \left(\frac{E}{L_j}\right) \tag{4.25}$$

$$\frac{\partial \psi_i}{\partial u_5} = \frac{-6.0}{\sigma_j^a} * \left(\frac{E}{L_j^2}\right) * c_j \tag{4.26}$$

$$\frac{\partial \psi_i}{\partial u_6} = \frac{2.0}{\sigma_j^a} * \left(\frac{E}{L_j}\right) * c_j \qquad (4.27)$$

$$\frac{\partial \psi_i}{\partial b} = \frac{E}{(2.0*\sigma^a*L_j^2)} * (6*u_2+4*L_j*u_3-6*u_5+2*L_j*u_6) \qquad (4.28)$$

4.2 Parallel Computation Tasks For Geometrically Nonlinear Design Sensitivity Analysis

4.2.1 Displacement Constraints

In the present work, the adjoint variable method is used to find the design sensitivity of the displacement with respect to the design variable. There are two approaches to obtain the design sensitivity information in a parallel vector computer environment. If the number of design variables is relatively greater than the number of processors, then each processor is assigned to one design variables which produces a good parallel algorithm. If the number of design variables is very small with respect to the number of processors, then all the processors will execute the analysis in parallel for only one design variable at a time. In this work, the first case is assumed, since where in general that is true for large-scale structural problems. The following steps will explain the procedure for obtaining the design sensitivity analysis for the displacement constraints in a parallel computer environment:

Presched do 10 I = 1 , Number of Design Variables

Find the derivative of the internal loads with respect to the i^{th} design variable.

10 End presched do

Presched do 20 I = 1 , Number of degrees-of-freedom

Use equation (4.10) to find λ

Use equation (4.11) and equation (4.3) to solve for $(d\psi/db)$

20 End presched do

From the above pseudo-Fortran statements, one can see there is no communication involved in the computational process.

4.2.1 Stress Constraints

The design sensitivity analysis (DSA) for stress will follow the same procedure as displacement constraints. The derivative of the internal stress with respect to the displacement is already calculated and the following procedure will be implemented for the stress DSA:

Presched do 30 I = 1 , Number of elements

Use equation (4.10) to find λ

Use equation (4.11) and equation (4.3) to solve for $(d\psi/db)$

30 End presched do

Very little communication is required in the above procedure for stress DSA and therefore, good speed up can be expected in a parallel computer environment.

4.3 Design Sensitivity Analysis and Verifications

In order to verify the answers obtained in the proposed parallel nonlinear DSA, the central finite difference method is used to compare the numerical accuracy. The derivative equation using the central finite difference method can be given as:

$$\frac{dU}{db_i} = \frac{U(b_i+h) + U(b_i-h)}{2*h} \tag{4.29}$$

Where dU/db_i can be represented as the change in the displacement with respect to the design variable b_i. $U(b_i+h)$ and $U(b_i-h)$ are the displacement vector evaluated using the perturbed design where h represents a small change to the design variable b_i. To compare the DSA accuracy using the finite difference (with 0.5% perturbation to the design variables) and the adjoint derivative approaches, a two dimensional frame structure (see Figure 3.3) have indicated very good agreements between the two approaches.

Example 4: Three-Dimensional Truss Structure.

Truss A given in Figure 3.2 is used as an example for the nonlinear DSA. From Table 4.1, it is clear that the minimum efficiency obtained for DSA is 98.8%. This good performance is expected since in the proposed DSA algorithm, very little communication is required. The total performance of the nonlinear structural analysis and the DSA is also good with an efficiency of 89.23% when eight processors were used.

Example 5: Two-Dimensional Frame Structure.

Frame A given in Figure 3.3 is used to illustrate the performance of the parallel-vector DSA algorithm. The results for the DSA is given in Table 4.2. The speed up obtained is also quite good where the minimum efficiency obtained is 80.46% when six processors were used. It can be observed that, the efficiency obtained by eight processors is higher than the case where six processors were used. The reason is because for the fixed number of elements and number of D.O.F., they are equally divided by eight processors (but not by six processors). Furthermore, it is also due to the fact that total DSA time is small in this example.

Example 6: Three-Dimensional Control-Structure Interaction (CSI) Model.

The results for the CSI model (refer to Figure 3.4) for the DSA is given in Table 4.3. For this example, only the displacement constraints were used. The efficiency obtained is 97.54% when four processors were used. The total time used to complete the DSA is relatively high since the total number of D.O.F is also large.

4.4 Some Remarks on Design Sensitivity Analysis

Design sensitivity analysis is an important component for formal optimization. The DSA process consumes a considerable amount of computer time. The evaluation of the design sensitivity analysis can be performed in parallel with very little communications required in a multi-processor computer environment. The speed up and the efficiency obtained was quite good since processor-communication is kept to the minimum.

The geometrically nonlinear design sensitivity analysis of the proposed procedure has been verified by comparing with the known solution in Ref. [Ryu et al.,1985]. Detail discussion of the numerical data in Ref. [Ryu et al.,1985] and the numerical comparison are given in Appendix B.

Table 4.1 DSA for Three Dimensional Truss (1872 D.O.F) with eight design variables.

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
T.NON	4.5877	2.4440	1.2411	0.8337	0.6427
T.DSA	23.9664	12.1658	6.1426	4.0006	3.0126
T.TNAS	29.7211	15.2640	8.0085	5.2540	4.0430
S.NON	----	1.8771	3.6965	5.5028	7.1382
S.DSA	----	1.9700	3.9017	5.9907	7.9554
S.TNAS	----	1.9471	3.7112	5.6569	7.3512
E.NON	100.0000	93.8600	92.4100	91.7100	89.2300
E.DSA	100.0000	98.5000	97.5400	99.8500	99.4400
E.TNAS	100.0000	97.3600	92.7800	94.2800	91.8900

Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 4.2 DSA for Two Dimensional Frame (880 D.O.F) with eight design variables.

	Number of Processors				
	1	2	4	6	8
NLS	2	2	2	2	2
T.NON	17.6700	9.1097	5.2560	3.8378	3.1410
T.DSA	5.0572	2.5392	1.2871	1.0476	0.7199
T.TNAS	23.0736	11.6319	6.8709	4.9139	4.1321
S.NON	----	1.9397	3.3619	4.6042	5.6256
S.DSA	----	1.9917	3.9291	4.8274	7.0249
S.TNAS	----	1.9836	3.3582	4.6956	5.5840
E.NON	100.0000	96.9900	84.0500	76.7400	70.3200
E.DSA	100.0000	99.5900	98.2300	80.4600	87.8100
E.TNAS	100.0000	99.1800	83.9600	78.2600	69.8000

Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

Table 4.3 DSA for CSI Structure (3096 D.O.F) with thirty two design variables.

	Number of Processors				
	1	2	4	6	8
NLS	4	4	4	4	4
T.NON	15.4768	8.1298	4.1068	2.9076	2.2580
T.DSA	54.7983	27.3963	13.7594	9.1838	6.9441
T.TNAS	70.6913	36.5261	18.2459	12.5074	9.2024
S.NON	----	1.9037	3.7686	5.3229	6.8542
S.DSA	----	2.0002	3.9826	5.9668	7.8913
S.TNAS	----	1.9354	3.8744	5.6520	7.6818
E.NON	100.0000	95.1900	94.2200	88.7200	85.6800
E.DSA	100.0000	100.0100	99.5700	99.4500	98.6400
E.TNAS	100.0000	96.7700	96.8600	94.2000	96.0200

Computer Used = Cray Y-MP (Reynolds)
Time Used = TSECND()
Tolerance = 1.0E-03

CHAPTER 5

SUMMARY AND CONCLUSIONS

A simple and efficient parallel algorithm for generating and assembling the structural stiffness matrix has been developed and tested in a non-dedicated computer environment on the Cray Y-MP. The new algorithm is based on a node-by-node assembly approach rather than the conventional element-by-element approach. The new approach circumvents with the processor synchronization problem associated with implementing the conventional approach on parallel machines. Good speed up factors were obtained even for small to medium structural examples. The new algorithm is general since there is no assumption made on the type of finite elements used.

For solving the geometrically nonlinear finite element problems, one concludes that the N-R shows the least time for completing the analysis in a parallel-vector computers environment. The speed up is quite good since the N-R method depends heavily on generating and solving the system of equations at each iteration.

The hybrid Choleski-PCG method has reduced significantly, the time spent to solve the linear system of equations as compared to the use of Choleski method alone. There is no extra overhead time to calculate a preconditioning matrix since the factorized matrix is reused in the next few iterations.

The new parallel algorithm for the S.O.R. method on shared memory computers does offer good parallel and vector speeds.

The proposed procedure for design sensitivity analysis of geometrically nonlinear finite element problems also offers a good speed up since very little processor-communication is required.

REFERENCES

Agarwal, T.K., Storaasli, O.O., Nguyen, D.T., "A Parallel-Vector Algorithm for Rapid Structural Analysis on High Performance Computers," Proceeding of the 31st AIAA/ASME/ASCE/AHS/SDM Conference, Long Beach, CA, (April 2-4 1990).

Arora, J.S., Introduction to optimum Design, McGraw-Hill, 1989.

Baddourah, M.A., Storaasli, O.O, Carmaona, E.A. and Nguyen, D.T., "Parallel and vector Procedures for System of Nonlinear of Equations," Presented at the ASCE Structures Congress, Baltimore, Maryland (April 30 - May 3,1990).

Bathe, K.J., Finite Element Procedures in Engineering Analysis, Prentice-Hall (1982).

Bathe, K.J., Wilson, E.L. and Iding R.H., "A Nonlinear Structural Analysis Program," Department of Civil Engineering, University of California, Berkely, California 97420.

Bathe, K.J. and Dvorkin E.N., "On the Automatic Solution of Nonlinear Finite Element Equations," Computers & structures, Volume 17, pp. 871-879 (1983).

Bathe, K.J. and Cimento, A.P., "Some Practical Procedures for the Solution of Nonlinear Finite Element Equations," Comp. Meth. Appl. Mech. Engng, Volume 22, pp. 59-85 (1980).

Belvin, W.K., Eliott, K.E., Bruner, A., Sulla, J., and Bailey, J., "The LaRC CSI Phase-O Evolutionary Model Tested: Design and Experimental Results," Proceedings of the Fourth Annual NASA/DOD Conference on Control/Structure Interaction Technology, Orlando, Florida, (November 5-7, 1990).

Chang, T.Y. and Padvan, J., "General Purpose nonlinear Finite Element Program," Struct. Mech. Software. Volume 3, pp. 79-99 (1980).

Chapra, S.C. and Canale, R.P., "Numerical Methods For Engineers With Computer Applications," McGraw-Hill (1985).

Chen, L.S., and Sun, C.T., "Parallel Processing Techniques For Finite Element Analysis of Nonlinear Large Truss Structures," Computers & Structures, Volume 31, No. 6, pp. 1023-1029, (1989).

Choi, K.K. and Santos, J.L.T, "Design Sensitivity Analysis of Non-Linear Structural Systems Part I: Theory," International Journal for Numerical Methods in Engineering, Volume 24, pp. 2039-2055 (1987).

Cook, R.D., Malkus, D.S., and Plesha, E., M.E., Concepts and Applications of Finite Element Analysis, Third Edition, John Wiley &sons (1989).

Crisfield, M.A., "A Faster Modified Newton-Raphson Iteration," Comp. Meth. Appl. Mech. Engng, Volume 20, pp. 267-278 (1979).

Gopalakruskna, H.S. and Greimann, L.F.,"Newton-Raphson Procedure For The Sensitivity of Nonlinear Structural Behavior," Computers and Structures, Volume 30, No. 6, pp. 1263-1273 (1988).

Jordan, H.F., Benton, M.S., Arenstrof, N.S., and Ramanan, A.V., "Force Users's Manual," Dept of Electrical and Computer Engineering, University of Colorado, Boulder, CO 80309.

Khot, N.S. and Kamat, M.P., "Minimum Weight Design of Structures with Geometric Nonlinear Behavior," Proceedings of the AIAA/ASME?ASCE/ASH 24th Structures, Structural Dynamics and Materials Conference, pp. 360-367 (May, 1983).

Law, K.H., " A Parallel Finite Element Solution Method," Computers & structures, Volume 23, pp. 845-858 (1986).

Matthies, H. and Strang, G., "The Solution of Nonlinear Finite Element Equations," International Journal For Numerical Methods in Engineering, Volume 14, pp. 1613-1626 (1979).

Nguyen, D.T., Storaasli, O.O., Carmona, E.A., Al-Nasra, M., Zhang, Y., Baddourah, M.A. and Agarwal, T.K., "Parallel-Vector Computation For Linear Structural Analysis and Nonlinear Unconstrained Optimization Problems," to appear in computing System in Engineering, An International Journal.

Ortega, J.M.,"Introduction to Parallel and Vector Solution of Linear Systems," Plenum Press, (1988).

Przemieniecki, J.S., Theory of Matrix Structural Analysis, McGraw-Hill, (1968).

Qin, J. and Nguyen, D.T., "A Parallel-Vector Lanczos Eigensolver for Structural Vibration Problem," Presented at the 4th International Conference on Recent Advances in Structural Dynamics, at The University os South Hampton, (July, 1991).

Riks, E., "The Application of Newton's Method to the Problem of Elastic Stability," Journal of Applied Mechanics, volume 39, pp 1060-1066 (1972).

Riks, E., "An Incremental Approach to the Solution of Snapping and Buckling Problems," International Journal of Solids Structure, Volume 15, pp 529-551 (1979).

Ryu, Y.S., Haririan, M., Wu, C.C. and Arora, J.S.,"Structural Design Sensitivity Analysis of Nonlinear Response," Computers and Structures, Volume 21, pp.245-255 (1985).

Storaasli, O.O., Nguyen, D.T. and Agarwal, T.K., "A parallel-Vector Algorithm For Rapid Structural Analysis on High-Performance Computers," NASA-TM 102614, NASA LaRC (April 1990).

Storaasli, O.O., Nguyen, D.T., and Agarwal, T.K., " The Parallel Solution of Large Scale Structural Analysis Problems on Supercomputers," proceedings AIAA/ASME/ASCE/AHS/ACS 30th SMD Conference, Mobile, Alabama (April 3-5,1989). Also appeared in the AIAA Journal, Volume 28, No. 7, pp. 1211-1216 (July 1990).

Tsay, J.J. and Arora, J.S., "Optimum Design of Nonlinear Structural With Path Dependent Response," Structural Optimization, Volume 1, pp. 203-213 (1989).

Utku, S., Melosh, R., and Islam M., "On Nonlinear Finite Element Analysis Single-, Multi-And Parallel-Processors," Computers & Structures, Volume 15, pp.39-47 (1982).

Vanderplaats, G.N., "Structural Optimization-Past, Present, and Future," AIAA J. 20, 992-1000 (1982).

Vanderplaats, G.N., "Structural Optimization-Past, Present, and Future," AIAA J. 20, 992-1000 (1982).

Wempner, G.A.,"Discrete Approximations Related to Nonlinear Theories of Solids," International Journal Solids Structure, Volume 7, pp 1581-1599 (1971).

Wen, R.K. and Rahimzadeh, J., "Nonlinear Elastic Frame Analysis by Finite Element," Journal of Structural Division, Volume 109, pp. 1952-1971 (1984).

Wu, C.C. and Arora, J.S., "Design Sensitivity Analysis and Optimization of Nonlinear Structural Response Using Incremental Procedure," AIAA Journal, Volume 25, pp. 1118-1125 (August 1987).

Zois, D., "Parallel Processing Techniques for FE Analysis: Stiffness, Loads and Stresses Evaluation," Computers & Structures, Volume 28, pp. 247-260 (1988).

APPENDICES

A. DESIGN SENSITIVITY ANALYSIS OUTPUT FOR TWO DIMENSIONAL FRAME WITH (18 D.O.F) USING FINITE DIFFERENCE METHOD AND THE ADJOINT METHOD.

Design variable No: 1

Displacement Constraints			Stress Constraints		
D.O.F	F.D.M.	Adjoint Method	Element	F.D.M	Adjoint Method
1	0.86802E-04	0.86791E-04	1	0.16653E+03	0.16651E+03
2	0.74571E-04	0.74565E-04	2	-0.12033E+03	-0.12031E+03
3	0.47687E-05	0.47680E-05	3	-0.29929E+02	-0.29926E+02
4	0.57397E-04	0.57388E-04	4	0.17366E+03	0.17364E+03
5	0.11108E-03	0.11107E-03	5	-0.57099E+03	-0.57088E+03
6	0.98420E-05	0.98405E-05	6	0.73797E+02	0.73792E+02
7	0.71102E-04	0.71095E-04	7	-0.42298E+02	-0.42285E+02
8	0.25101E-03	0.25096E-03	8	-0.51770E+02	-0.51769E+02
9	0.12433E-04	0.12429E-04	9	-0.16733E+03	-0.16730E+03
10	0.84566E-04	0.84554E-04	10	-0.29092E+03	-0.29089E+03
11	0.49108E-04	0.49103E-04	11	0.65225E+02	0.65204E+02
12	-0.16332E-05	-0.16328E-05	12	-0.23525E+02	-0.23530E+02
13	0.91168E-04	0.91152E-04	13	-0.40112E+03	-0.40110E+03
14	0.14081E-03	0.14080E-03	14	-0.19853E+03	-0.19852E+03
15	-0.63276E-05	-0.63278E-05	15	0.14928E+03	0.14926E+03
16	0.75599E-04	0.75585E-04	16	0.31156E+02	0.31120E+02
17	0.63744E-04	0.63740E-04	17	-0.51991E+03	-0.51985E+03
18	-0.91905E-05	-0.91887E-05	18	-0.33535E+03	-0.33530E+03

Design variable No: 2

Displacement Constraints			Stress Constraints		
D.O.F	F.D.M.	Adjoint Method	Element	F.D.M	Adjoint Method
1	-0.19499E-03	-0.19400E-03	1	0.91427E+02	0.90966E+02
2	-0.88626E-05	-0.88169E-05	2	0.12319E+03	0.12254E+03
3	0.14638E-04	0.14564E-04	3	-0.12045E+03	-0.11983E+03
4	-0.18591E-03	-0.18496E-03	4	-0.25468E+03	-0.25338E+03
5	0.64756E-04	0.64426E-04	5	0.42235E+02	0.42022E+02
6	0.10050E-04	0.99986E-05	6	-0.10867E+03	-0.10810E+03
7	-0.23860E-03	-0.23737E-03	7	0.92210E+02	0.91736E+02
8	0.21854E-03	0.21742E-03	8	0.10160E+03	0.10108E+03
9	0.16801E-04	0.16715E-04	9	-0.76632E+02	-0.76249E+02
10	-0.36451E-04	-0.36262E-04	10	0.17104E+03	0.17017E+03
11	-0.21030E-04	-0.20923E-04	11	-0.36118E+02	-0.35936E+02
12	0.76420E-05	0.76024E-05	12	0.14269E+03	0.14196E+03
13	-0.57167E-04	-0.56876E-04	13	-0.41832E+02	-0.41627E+02
14	0.24935E-04	0.24809E-04	14	-0.76753E+03	-0.76362E+03
15	0.11904E-04	0.11843E-04	15	0.61580E+01	0.61299E+01
16	-0.87266E-04	-0.86821E-04	16	-0.68253E+02	-0.67911E+02
17	0.22286E-03	0.22172E-03	17	0.11653E+03	0.11593E+03
18	0.15503E-04	0.15423E-04	18	-0.32304E+03	-0.32139E+03

Design variable No: 3

Displacement Constraints			Stress Constraints		
D.O.F	F.D.M.	Adjoint Method	Element	F.D.M	Adjoint Method
1	0.14230E-04	0.14220E-04	1	0.96061E+02	0.96045E+02
2	0.15014E-03	0.15011E-03	2	0.10085E+02	0.10091E+02
3	-0.18291E-04	-0.18289E-04	3	-0.59784E+03	-0.59776E+03
4	-0.14957E-04	-0.14963E-04	4	-0.57829E+02	-0.57836E+02
5	0.30918E-04	0.30918E-04	5	0.35778E+02	0.35767E+02
6	-0.43364E-06	-0.43289E-06	6	-0.24567E+03	-0.24565E+03
7	-0.13527E-04	-0.13532E-04	7	0.83674E+02	0.83664E+02
8	0.57917E-05	0.57932E-05	8	-0.42095E+02	-0.42077E+02
9	-0.45549E-05	-0.45545E-05	9	-0.12378E+02	-0.12375E+02
10	-0.34222E-04	-0.34224E-04	10	-0.19810E+02	-0.19811E+02
11	0.21080E-04	0.21080E-04	11	0.19600E+03	0.19599E+03
12	0.35413E-05	0.35413E-05	12	-0.13175E+03	-0.13176E+03
13	-0.15934E-04	-0.15934E-04	13	-0.12010E+03	-0.12009E+03
14	0.21279E-04	0.21277E-04	14	-0.53693E+02	-0.53692E+02
15	-0.63024E-06	-0.63006E-06	15	-0.73521E+02	-0.73526E+02
16	-0.52126E-04	-0.52124E-04	16	-0.92424E+02	-0.92421E+02
17	0.17401E-05	0.17405E-05	17	-0.44457E+02	-0.44451E+02
18	0.45252E-05	0.45252E-05	18	0.95635E+02	0.95633E+02

Design variable No: 4

Displacement Constraints			Stress Constraints		
D.O.F	F.D.M.	Adjoint Method	Element	F.D.M	Adjoint Method
1	0.60151E-04	0.60429E-04	1	-0.10375E+03	-0.10426E+03
2	0.17397E-03	0.17482E-03	2	0.84730E+02	0.85132E+02
3	-0.26219E-05	-0.26341E-05	3	-0.14698E+02	-0.14773E+02
4	0.58591E-04	0.58862E-04	4	-0.44797E+03	-0.45013E+03
5	0.16318E-03	0.16397E-03	5	0.85879E+01	0.86239E+01
6	-0.13185E-04	-0.13250E-04	6	0.10552E+03	0.10604E+03
7	0.47955E-04	0.48176E-04	7	-0.50215E+02	-0.50468E+02
8	0.10961E-04	0.11020E-04	8	-0.24659E+03	-0.24779E+03
9	-0.12844E-04	-0.12906E-04	9	-0.79997E+02	-0.80374E+02
10	0.22601E-04	0.22705E-04	10	0.32143E+03	0.32302E+03
11	0.17311E-03	0.17395E-03	11	0.68993E+02	0.69345E+02
12	-0.76320E-05	-0.76674E-05	12	-0.91238E+03	-0.91680E+03
13	-0.37146E-04	-0.37337E-04	13	-0.34658E+03	-0.34830E+03
14	0.68096E-04	0.68434E-04	14	-0.19089E+03	-0.19185E+03
15	-0.14724E-05	-0.14786E-05	15	-0.19917E+03	-0.20014E+03
16	-0.38250E-04	-0.38445E-04	16	-0.24106E+03	-0.24226E+03
17	0.18692E-04	0.18787E-04	17	-0.15107E+03	-0.15180E+03
18	-0.28459E-05	-0.28593E-05	18	-0.37258E+02	-0.37437E+02

B. INPUT DATA PREPARATION FOR THE PARALLEL-VECTOR COMPUTER CODE NONDSA.FRC

I. HEADING CARD (12A6)

(1)	1-72	HED(12)	Enter the heading information.

II. MASTER CONTROL CARD I (3F10.2)

(1)	1-10	TOL1	Tolerance used for nonlinear analysis.
(2)	11-20	TOL2	Tolerance used for iterative solvers.
(3)	21-30	W	Omega used in S.O.R.

III. MASTER CONTROL CARD II (9I5)

(1)	1-5	NNODE	Number of nodes.
(2)	5-10	1	Not used
(3)	11-15	1	Not used
(4)	15-20	0	Not used
(5)	21-25	MODE	Type of analysis: 1 Linear. 2 Nonlinear.
(6)	25-30	METHOD	Method for nonlinear analysis: 1 Piecewise Linear approximation.

			2 Newton-Raphson method.
			3 BFGS method.
			4 Modified Newton-Raphson Method.
(7)	31-35	NLD	Number of loading steps.
(8)	36-40	ICHEK	Printing control:
			0 Only final answers printed.
			1 Used for debugging.
(9)	41-45	ISWTCH	If ≥ NLD, Choleski used in all steps.
			if < NLD, Iterative methods used after ISWTCH.
(10)	46-50	IMETHD	Methods for iterativesolver:
			1 S.O.R. Method
			2 P.C.G. Method.

IV. NODAL POINT DATA (I5,6I5,3F10.2,I5,F10.0)

(1)	1-5	N	Node number.
(2)			Boundary code. 0 = Free, 1 = Fixed.
	6-10	ID(1,N)	X-translation boundary code.
	11-15	ID(2,N)	Y-translation boundary code.
	16-20	ID(3,N)	Z-translation boundary code.
	21-25	ID(4,N)	X-rotation boundary code.
	26-30	ID(5,N)	Y-rotation boundary code.
	31-35	ID(5,N)	Z-rotation boundary code.

(3)	36-45	X(N)	X-coordinate
	46-55	Y(N)	Y-coordinate
	56-65	Z(N)	Z-coordinate
(4)	66-70	KN	0 (not used)
(5)	70-80	TEMP(N)	0. (not used)

V. ELEMENT DATA

TYPE 1 - THREE DIMENSIONAL TRUSS ELEMENT

A. Control Card (9I5)

(1)	1-5	IFLAG	The number 1
(2)	6-10	NE	Total number of truss elements.
(3)	11-15	NELMP	Number of material property.
(4)	16-20		0 (not used).
(5)	21-25		0 (not used).
(6)	26-30	NSTORY	Number of stories.
(7)	31-35	NSTORY	Number of bays.
(8)	36-40	IFIND	Finite difference check. 1 Yes, 0 No.
(9)	41-45	ISEN	Design sensitivity analysis. 1 Yes, 0 No.

B. Material Property Cards (I5,5F10.0)

1-5	MID	Material identification number.
6-10	E	Modulus of elasticity.
16-25		0.0 (not used).
26-35		0.0 (not used).

36-45	A	Cross sectional area.
46-55		0.0 (not used).

C. Element Load Factors: (4F10.0) Four cards.

Card 1: Multiplier of gravity load in the +X direction

1-10	0.0 (not used).
11-20	0.0 (not used).
21-30	0.0 (not used).
31-40	0.0 (not used).

Card 2: As above for gravity load in the +Y direction (not used).

Card 3: As above for gravity load in the +Z direction (not used).

Card 4: As above (not used).

D. Element Data Cards (5I5,F10.0,I5)

1-5	NN	Element number.
6-10	NI	Node number I.
11-15	NJ	Node number J.
16-20	MIDN	Material property number.
21-30		0 (not used).
31-35		0 (not used).

TYPE 2 - THREE DIMENSIONAL BEAM ELEMENT

A. Control Card (9I5)

(1)	1-5	IFLAG	The number 2
(2)	6-10	NE	Total number of beam elements.
(3)	11-15	NELMP	Number of element material property cards.

(4)	16-20		0 (not used).
(5)	21-25		Number of material property cards.
(6)	26-30	NSTORY	Number of stories.
(7)	31-35	NSTORY	Number of bays.
(8)	36-40	IFIND	Finite difference check. 1 Yes, 0 No.
(9)	41-45	ISEN	Design sensitivity analysis. 1 Yes, 0 No.

B. Material Property Cards (I5,4F10.0)

1-5	MID	Material identification number.
6-10	E	Modulus of elasticity.
16-25	V	Poisson's ratio.
26-35		0.0 (not used).
36-45		0.0 (not used).

B. Element Property Cards (I5,5F10.0)

1-5	GID	Geometric identification card.
6-10	AX	Axial Area.
16-25	AY	Shear area associated with local 2-direction.
26-35	AZ	Shear area associated with local 3-direction.
26-35	IZ	Torsional inertia.
36-45	IX	Flexural inertia about local 2-direction.
46-55	IY	Flexural inertia about local 3-axis.

C. Element Load Factors: (4F10.0) Four cards.

Card 1: Multiplier of gravity load in the +X direction

1-10	0.0 (not used).

11-20		0.0 (not used).
21-30		0.0 (not used).
31-40		0.0 (not used).

Card 2: As above for gravity load in the +Y direction (not used).

Card 3: As above for gravity load in the +Z direction (not used).

D. Element Data Cards (10I5,2I6,I8)

1-5	NN	Element number.
6-10	NI	Node number I.
11-15	NJ	Node number J.
16-20	NK	NK any nodal point lies in the 1-2 plane.
21-25	MIDN	Material property number.
26-30	IGIDN	Geometric property number.
31-35		0 (not used).
35-40		0 (not used).
41-45		0 (not used).
46-50		0 (not used).
51-56		0 (not used).
57-62		0 (not used).
63-70		0 (not used).

VI. CONCENTRATED LOAD DATA (2I5,6F10.4)

(1)	1-5	NODEN	Nodal point number.
(2)	6-10	L	EQ.1, static analysis.

11-20	PX	X-direction force.
21-30	PY	Y-direction force.
31-40	PZ	Z-direction force.
41-50	MX	X-axis moment.
51-60	MY	Y-axis moment.
61-70	MZ	Z-axis moment.

Note: If NNODE .EQ. 0, the program will go to the next read statement.

VII. SENSITIVITY ANALYSIS DATA

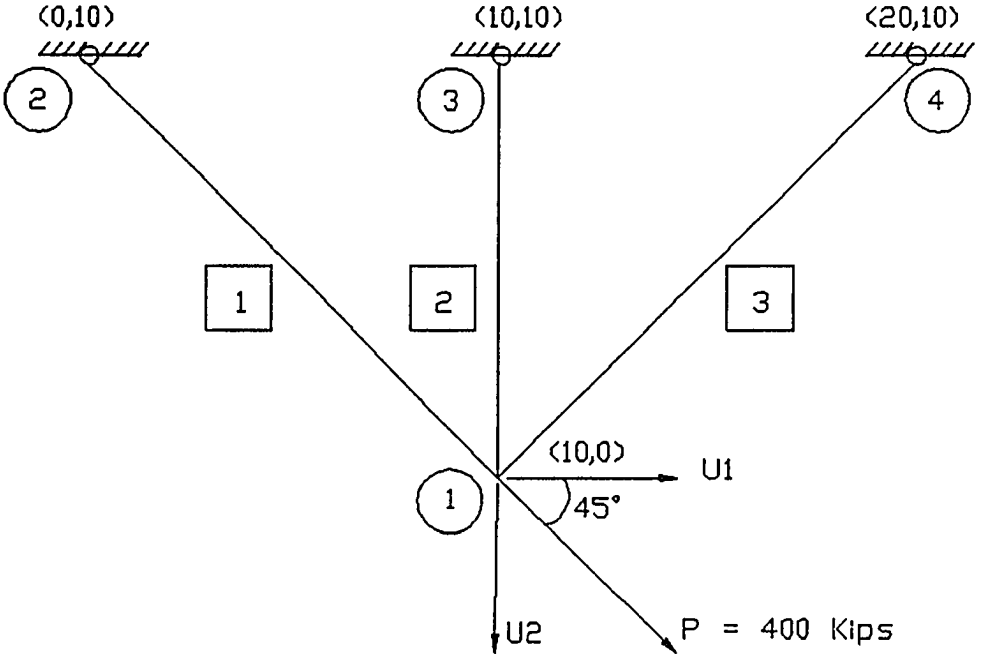
A. Sensitivity Control Card (3I5)

(1)	1-5	NDV	Number of design variable.
	5-10	NDC	Number of displacements constraints.
	11-15	NSC	Number of stresses constraints.

B. Allowable Data Card (2F10.2)

1-10	ADISP	Allowable displacement.
11-20	ASTRESS	Allowable stress.

C. INPUT DATA FILE AND NUMERICAL VERIFICATION FOR NONLINEAR DESIGN SENSITIVITY ANALYSIS OF THE THREE BAR TRUSS



b1 = 0.10 IN^2
b2 = 0.20 IN^2
b3 = 0.20 IN^2
E = 10000 KSI

```
** THREE BAR TRUSS DATA FILE USING NONDSA.FRC PROGRAM ******
1.0E-03      1.0E-03  1.00
   4    1    1    0    1    2    2    1  40   1
   1    0    0    1    1    1    1     10.0       0.0      0.000    0         0.
   2    1    1    1    1    1    1      0.0       10.0     0.000    0         0.
   3    1    1    1    1    1    1     10.0       10.0     0.000    0         0.
   4    1    1    1    1    1    1     20.0       10.0     0.000    0         0.
   1    3    3    0    0    0    0    1    1    0    0    0    0    0
   1     10000         1.         1.       0.10
   2     10000         1.         1.       0.20
   3     10000         1.         1.       0.30
      1.          0.          0.          0.
      0.          1.          0.          0.
      0.          0.          1.          0.
      0.          0.          0.          1.
   1    1    2    1      0.00
   2    1    3    2      0.00
   3    1    4    3      0.00
   1    1   282.00   -282.00   00000.0    0.0000    0.0000    0.0000
   0    1   0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
   3    2    3
    1.20        0.5    15000.0
```

Results:

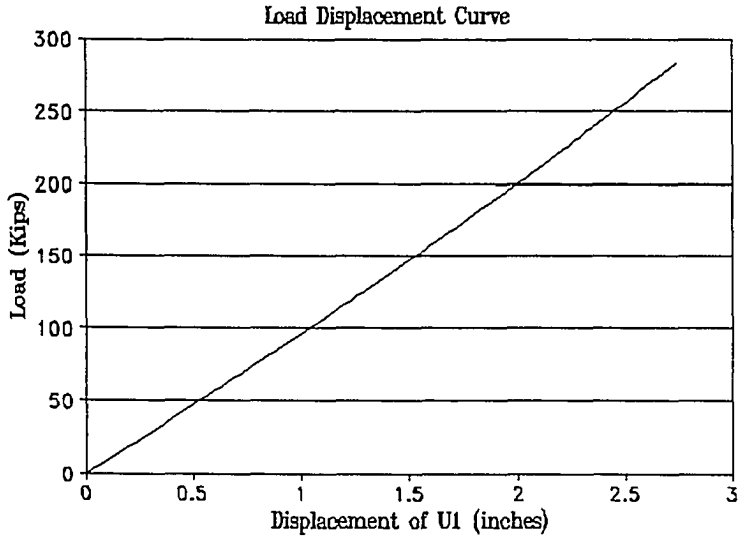
Displacements:

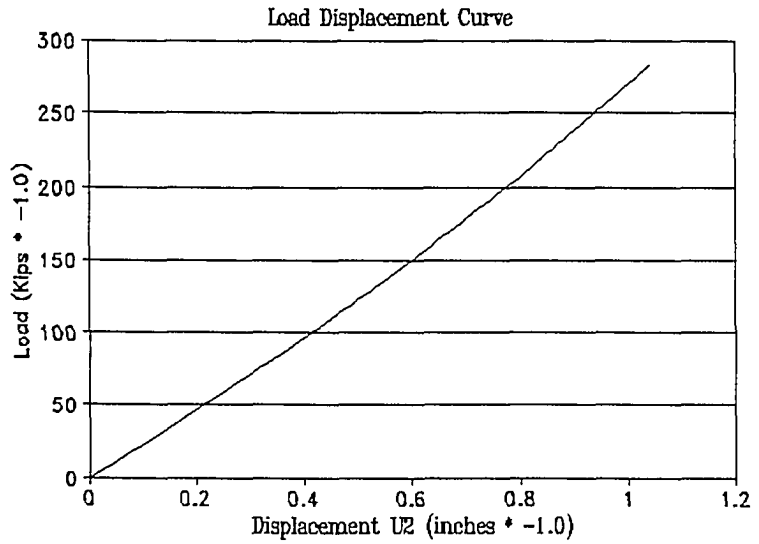
	U1	U2
Present Work	2.73392	-1.03987
Ryu's Result (Ref. [Ryu et al. 1985])	2.73391	-1.03987

Sensitivity Analysis (Displacement Constraints):

	Present Work	Ryu's Work
$d\psi_1 / db_1$	-6844.480	-6846.114
$d\psi_2 / db_1$	-802.456	-804.045
$d\psi_1 / db_2$	-1545.363	-1545.555
$d\psi_2 / db_2$	-858.256	-858.298
$d\psi_1 / db_3$	-1712.404	-1710.228
$d\psi_2 / db_3$	354.633	354.582

Load Displacement Curve For The Three Bar Truss.

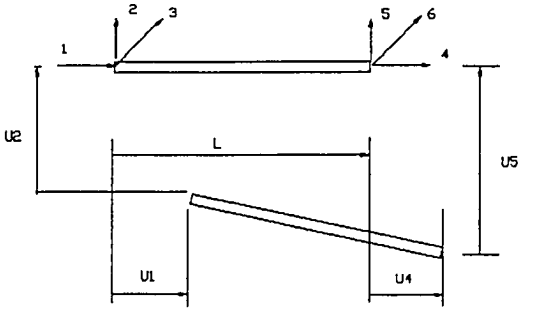




D. LINEAR AND NONLINEAR STIFFNESS MATRIX

FOR BAR AND BEAM ELEMENT

BAR ELEMENT



Only the non zero entries in the element stiffness matrix are given.

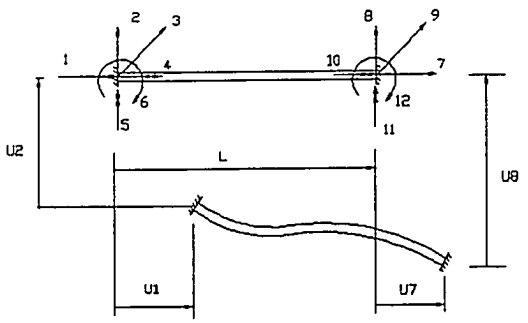
$K_{1,1} = K_{4,4} = E A / L + (F / L)$

$K_{4,1} = -E A / L + (F / L)$

$K_{4,1} = K_{5,2} = K_{6,3} = (F / L)$

$F = E A (U4 - U1) / L$

BEAM ELEMENT



Only the non zero entries in the element stiffness matrix are given.

$K_{1,1} = K_{7,7} = -K_{1,6} = E A / L$

$K_{2,2} = K_{8,8} = -K_{2,8} = 12 E I_z / L^3 + (6 F / 5 L)$

$K_{2,6} = K_{2,12} = 6 E I_z / L^3 + (F / 10)$

$K_{3,3} = K_{9,9} = -K_{3,9} = 12 E I_y / L^3 + (6 F / 5 L)$

$K_{3,5} = K_{3,12} = -6 E I_y / L^2$

$K_{4,4} = K_{10,10} = -K_{4,10} = G J / L$

$K_{5,5} = K_{11,11} = 4 E I_y / L + (2 F L / 15)$

$K_{5,8} = 6 E I_y / L^2$

$K_{5,10} = 2 E I_y / L$

$K_{6,6} = K_{12,12} = 4 E I_z / L + (2 F L / 15)$

$K_{6,8} = -6 E I_z / L^2$

$K_{6,12} = 2 E I_z / L + (F L / 30)$

$K_{8,12} = -6 E I_z / L^3$

$K_{9,11} = 6 E I_y / L^2 + (F L / 30)$

$F = E A (U7 - U1) / L$

E. LISTING OF THE PARALLEL COMPUTER CODE NONDSA.FRC

```
C This program is revised on January 1 1991     BY: Majdi Baddourah
C NONDSA.FRC is a linear and nonlinear finite element analysis and
C desigen sensitivity analysis for:
C     A- 3-D Truss
C     B- 2-D Frame
C
C Methods included in this program:
C Newton Raphson method.
C Modified Newton Raphson method.
C BFGS Method.
C Peicewise linear approximation.
C Force is the language used in this program.
C Cray Y-MP and Cray 2 version.
C ***************************************************************
C ********  Some Important Defenitions Used In MAIN.FRC  ********
C ***************************************************************
C DISPA  = Allawable displacement in DSA.
C SIGMAA = Allawable stresses in DSA.
C AREAL  = Lower area allowed in DSA.
C NDV    = Number of design variable.
C NDC    = Number of displacement constraints.
C NTDC   = Number of total design constraints.
C NSC    = Number of stress constriants in DSA.
C EPS1   = Nonlinear analysis tolerance ( unbalanced forces ).
C EPS2   = Iterative procedures tolerance. (|| [A]{X}-{B} || = EPS2
C NNODE  = Number of nodes.
C NELMNT = Number of elements.
C NELTYP = Number of element type.
C ICHKSN = 1  Check sensitivity using finite difference scheme.
C          0  Disable finite difference checking.
C METHOD = Type of nonlinear method used in the nonlinear analysis.
C NLFLAG = 1  Linear analysis.
C          2  Nonlinear analysis.
C NLD    = Number of loading steps.
C NDPN   = Number of D.O.F. per node.
C ICHK   = Debuging purpose.
C IW     = Writting unit.
C IW1    = Writing unit for debuging.
C IR     = Reading unit.
C LOOPL  = Level of loop unrooling.
C NOIIT  = Number of maximum iterations in BFGS.
C MAXNIT = Maximum nonlinear iteration allowed.
C -------------------------------------------------------------

      Force MAB of NP ident ME
      Shared REAL TT1(16),TT2(16),TT3(16),TT4(16),TT5(16),TT6(16)
      Shared  REAL  A(900000)
      Shared  INTEGER KA(200000)
      Shared COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      Shared COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      Shared   COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      Shared   COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      Shared COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      Shared COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,
     & KL20
      Shared COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      Shared COMMON/ALLI8/K40,K41,K42,K43,K44
      Shared COMMON/ALLI9/IOPT,K7F,K14F,K16F
      Shared COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,
     & NOMP
      Shared COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      Shared COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
      SharedCOMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,
     & K33,K34
      SharedCOMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,
     & MALY,K35
      Shared LOGICAL TYPE2
      Shared INTEGER MOPT,NOOCT
      Externf NEWTON,MNEWTN,BFGS,INCRMA,OPTIMT,BINCRM
      CHARACTER HED*72
      End declarations
      KLAST = 900000
```

```
      MLAST = 200000
      CALL CPUTIM(TT5(ME))
      Barrier
      JSTOP = 0
      IW = 6
      IW1  = 6
      IR = 5
      MESTOP = 0
      NOIIT = 8
      LOOPL = 8
      READ(IR,1000) HED
      READ(IR,*) EPS1,EPS2,W
      WRITE(IW,1000) HED
      WRITE(IW,*)
      READ(IR,1001) NNODE,NELTYP,LL,NF,NDYN,MODEX,NAD,
     & KEQB,ISWTCH,ISOR
      WRITE(IW,115)NNODE,NELTYP
      NLD = NAD
      METHOD = MODEX
      NLFLAG = NDYN
      ICHK   = KEQB
      ICHKSN = 0
      MAXNOP = 1
      MFINDJ = 1
      MFINDL = 1
      IF(NLFLAG .EQ. 1 ) THEN
      NLD = 1
      METHOD = 0
      ENDIF
      IF ( NLD .EQ. 0 ) NLD = 1
      WRITE(IW,*) ' Number of Load Division  = ', NLD
      WRITE(IW,*) ' METHOD                   = ', METHOD
      WRITE(IW,*) ' METHOD = 1               INCRIMENTAL APPROCH '
      WRITE(IW,*) ' METHOD = 2               NEWTON-RAPHSON METHOD'
      WRITE(IW,*) ' METHOD = 3               BFGS METHOD'
      WRITE(IW,*) ' METHOD = 4               MODIFIDE NEWTON METHOD'
      WRITE(IW,*) ' Problem Type             = ', NLFLAG
      WRITE(IW,*) '          1      LINEAR'
      WRITE(IW,*) '          2      NONLINEAR'
      K1  = 1
      K2  = NNODE + K1
      K3  = NNODE + K2
      K1A = NNODE + K3
      K2A = NNODE + K1A
      K3A = NNODE + K2A
      KL3 = NNODE + K3A
      M1  =  1
      ML1 =  NNODE * 6 + M1
      IFLAG = 1
      IF( METHOD .EQ. 1 ) THEN
      MAXNIT = 1
      ELSE
      MAXNIT = 30
      ENDIF
C---- READ IN NODAL POINTS
      CALL READND (IR,IW,NNODE,A(K1),A(K2),A(K3),KA(M1),NTDOF,ICHK)
      WRITE(IW,*)'  NUMBER OF ACTIVE DEGREES OF FREEDOM =',NTDOF
      IF( ICHK .EQ. 1) THEN
      WRITE(IW1,*)
      WRITE(IW1,*)' SUBROUTINE READND   IS COMPLETED '
      ENDIF
      READ(IR,1002) JFLAG,NELMNT,NC1,NC2,NC3,NC4,NC5,NC6,NC7
      NSTORY = NC4
      NBAY   = NC5
      ICHKSN = NC6
      MOPT   = NC7
      IF( JFLAG .EQ. 1 ) THEN
      NOMP = NC1
      NDPN = 3
      NNPE = 2
      NDPE = NNPE * NDPN
      NFD = NDPN * NNODE - NTDOF
      NTD = NTDOF
      NOMP = NC1
      WRITE(IW,125) NELMNT,NOMP
      K4  =  KL3
      K5  = NOMP +  K4
      K6  = NOMP + K5
      K7  = NOMP + K6
      K7F = NOMP + K7
      K8  = NOMP + K7F
      KL8 = NOMP + K8
      MI  = ML1
      ML  = MI + NELMNT * NNPE
      M2  =  ML
      ML2 = NELMNT + M2
      M3 = ML2
      K10 = KL8
```

```
      K10A = K10 + NELMNT
      K11 = NELMNT + K10A
      K12 = NELMNT + K11
      K13 = NELMNT + K12
      KL13 = NELMNT + K13
      CALL READTS (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(K6),A(K7),
     & A(K8),KA(MI),A(K10),A(K10A),A(K11),A(K12),A(K13),A(K1),A(K2),
     & A(K3),KA(M2) ,ICHK)
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW1,*)' SUBROUTINE READTS   IS COMPLETED '
      ENDIF
      ELSEIF( JFLAG .EQ. 2 ) THEN
      NOEP   = NC1
      NOFEFS = NC2
      NOMP   = NC3
      NDPN = 6
      NNPE = 2
      NDPE = NNPE * NDPN
      NFD = NDPN * NNODE - NTDOF
      NTD = NTDOF
      WRITE(IW,135) NELMNT,NOMP
      K4  =  KL3
      K5  = NOMP +  K4
      K6  = NOMP + K5
      K7  = NOMP + K6
      K8A = NOMP + K7
      K8B = NOEP + K8A
      K8C = NOEP + K8B
      K8I1 = NOEP + K8C
      K8I2 = NOEP + K8I1
      K8I3=NOEP + K8I2
      KL8 = NOEP + K8I3
      MI  = ML1
      ML  = MI + NELMNT * NNPE
      M2  =  ML
      ML2 = NELMNT + M2
      M2A = ML2
      M2B = NELMNT + M2A
      M2C = 4 * NELMNT + M2B
      ML2 = 2 * NELMNT + M2C
      M3 = ML2
      K10 = KL8
      K10A = NELMNT + K10
      K11 = NELMNT + K10A
      K12 = 9 * NELMNT + K11
      K13 = 6  + K12
      KL13 =  K13
      CALL       READBM (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(k6),
     & A(K7),
     & NOEP,A(K8A),A(K8B),A(K8C),A(K8I1),A(K8I2),A(K8I3) ,
     & A(K12),A(K1),A(K2),A(K3),A(K10),A(K10A),A(K11),
     & KA(MI),KA(M2),KA(M2A),KA(M2B),KA(M2C),NOFEFS,ICHK)
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW1,*)' SUBROUTINE READBM    IS COMPLETED '
      ENDIF
C ---
      ENDIF
      MAXJT = 0
      DO 10 LINDA = 1 , NNPE
      CALL      MAXJNE (IR,IW,NELMNT,NNODE,KA(MI),MAXJTI,KA(M3),
     & NNPE,LINDA,ICHK )
      IF(MAXJTI .GT. MAXJT ) MAXJT = MAXJTI
10    CONTINUE
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW1,*)' SUBROUTINE MAXJNE   IS COMPLETED '
      ENDIF
      WRITE(IW,*) ' MAX MEMBERS AT ONE JOINT =',MAXJT
      IF( NNPE .EQ. 2 ) THEN
      WRITE(IW,*) ' Number of Node per ELEMNT =',NNPE
      M3  = ML2
      M3A = M3 + NNODE * MAXJT
      M4  = M3A + NNODE * MAXJT
      M4A = NNODE + M4
      ML4 = NNODE + M4A
      CALL NEWID (IR,IW,NELMNT,NNODE,KA(MI),MAXJT,KA(M3) ,KA(M4) ,NNPE,
     & 1, ICHK)
      CALINEWID(IR,IW,NELMNT,NNODE,KA(MI),MAXJT,KA(M3A),KA(M4A),NNPE,
     & 2,ICHK)
      ENDIF
      M5  = ML4
      M6  = NTD  + M5
      M7  = NTD + 1 + M6
      ML7 = NTD + M7
      WRITE(6,*)' MAX NUMBER OF INTEGER  STORAGE = ',ML7
      CALL       COLH  (IR,IW,NNODE,NELMNT,NTD,KA(M1),KA(M5),KA(M6),
     & NTRMS,NDPE,KA(MI),NNPE,ICHK,NDPN )
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW1,*)' SUBROUTINE COLH      IS COMPLETED '
```

```
      ENDIF
      CALL     ROWLNG (IR,IW,NTD,KA(M7),KA(M6),NTRMS,NNODE,NELMNT,
     & LOOPL,NDPE,KA(M1),KA(M5),KA(MI),NNPE,ICHK )
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW1,*)' SUBROUTINE ROWLNG   IS COMPLETED '
      ENDIF
      K14 = KL13
      K14A = NTD + K14
      K14B = NTD + K14A
      K14C = NTD + K14B
      K14D = NTD + K14C
      K14F = NTD + K14D
      K15  = NTD + K14F
      K16  = NTD + K15
      K16F = NELMNT + K16
      K17  = NELMNT + K16F
      K18  = NTD + K17
      KL18 = NELMNT + K18
      K20 = KL18
      KL20 = NTRMS + K20
      K31  = KL20
      DO 30 IMAD = K14 , KL20
      A(IMAD) = 0.0
30    CONTINUE
      DO 31 IMAD = K14A, K14B + 1
31    A(IMAD) = .50
      JNLD = NLD
      CALL PLOAD (IR,IW,NNODE,A(K14),KA(M1),NTDOF,JNLD,ICHK,A(K17))
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW1,*)' SUBROUTINE PLOAD    IS COMPLETED '
      ENDIF
      READ(IR,1009)NDV,NDC,NSC
      READ(IR,1012)AREAL,DISPA,SIGMAA
      BL = DISPA
      NTDC = NDC + NSC
      K32 = K31 + NTD * NDV
      K32A= K32 + NDV * NTDC
      KL32 = K32A + NDV * NTDC
      IF( JFLAG .EQ. 1 ) THEN
      K40  = KL32
      ELSEIF( JFLAG .EQ. 2 ) THEN
      K33 = KL32
      K34 = K33 + NOEP
      K35 = K34 + NOEP
      KL35 = K35
      K40 = KL35
      ENDIF
      K41  = NOIIT * NTD + K40
      K42  = NOIIT * NTD + K41
      KL42 = NTD + K42
      WRITE(6,*)' MAX NUMBER OF REAL  STORAGE = ',KL42
      IF( KL42 .GT. KLAST ) THEN
      WRITE(6,*)' MAX NUMBER OF STORAGE EXEEDED ALLOWED'
      ENDIF
      WRITE(IW,*) ' Number Of Elements = ',NELMNT
      WRITE(IW,*) ' Number Of Nodes = ',NNODE
      WRITE(IW,*) ' Number of D.O.F Per Element = ',NDPE
      WRITE(IW,*) ' Number of active D.O.F = ',NTD
      WRITE(IW,*) ' Number of Terms = ',NTRMS
      WRITE(IW,*) ' Number of Storys = ',NSTORY
      WRITE(IW,*) ' Number of Bays   = ',NBAY
      WRITE(IW,*)
      WRITE(IW,*)'  NUMBER OF DESIGN VARIABLE = ',NDV
      WRITE(IW,*)'  NUMBER OF DISPLACEMENT CONSTRAINTS = ',NDC
      WRITE(IW,*)'  NUMBER OF STRESS CONSTRAINTS = ',NSC
      WRITE(IW,*)'  NUMBER OF TOTAL DESIGN CONSTRAINTS = ',NTDC
      IF( ICHKSN .EQ. 1 ) THEN
      MFINDJ = 3
      MFINDL = NDV
      MAXNOP = 1
      ENDIF
      WRITE(IW,*)
      WRITE(IW,*)'  Max Number Of Optimization Iterations =',MAXNOP
      WRITE(IW,*)'  Max Number Of Finite Diffrence    L   =',MFINDL
      WRITE(IW,*)'  Max Number Of Finite Diffrence    J   =',MFINDJ
      End barrier
      Barrier
      End barrier
C ... Optimization Do Loop Starts at The Next Statment.
      MAWYA = 1
      MUNA  = 1
      MALY  = 1
C     DO 60 MAWYA = 1 , MAXNOP
      Barrier
      WRITE(IW,*)
      WRITE(IW,*)'          ************************************'
      WRITE(IW,*)'           OPTIMIZATION ITERATION NO: ',MAWYA
      WRITE(IW,*)'          ************************************'
```

```
      DO 50 MUNA  = 1 , MFINDL
      WRITE(IW,*)'          OPTIMIZATIONL ITERATION NO: ',MUNA
      DO 40 MALY  = 1 , MFINDJ
      WRITE(IW,*)'          OPTIMIZATIONJ ITERATION NO: ',IOPT
      End Barrier
      Barrier
      End Barrier
      CALL CPUTIM(TT1(ME))
C ---- Linear Analysis.
      IF(NLFLAG .EQ. 1 ) THEN
      CALL LINEAR (A,KA)
      ENDIF
C ---- Nonlinear Analysis.
      IF(NLFLAG .EQ. 2 ) THEN
      IF(METHOD .EQ. 1) THEN
      Forcecall BINCRM (A,KA,W)
      ELSEIF(METHOD .EQ. 2) THEN
      Forcecall NEWTON (A,KA,W,NOOCT)
      ELSEIF( METHOD .EQ. 3 ) THEN
      Forcecall BFGS(A,KA,W,NOOCT)
      ELSEIF( METHOD .EQ. 4 ) THEN
      Forcecall MNEWTN(A,KA,W,NOOCT)
      ENDIF
      ENDIF
      CALL CPUTIM(TT2(ME))
C ---- Optimization Analysis.
      IF( MOPT .EQ. 1 ) THEN
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall OPTIMT (A,KA)
      ELSEIF( JFLAG .EQ. 2 ) THEN
      Forcecall BOPTIM (A,KA)
      ENDIF
      ENDIF
40    CONTINUE
50    CONTINUE
60    CONTINUE

      CALL CPUTIM(TT6(ME))
      Barrier
999   CONTINUE
      TIMENL = 0.0
      OPTIME = 0.0
      TIME = 0.0
      DO 202 I = 1 , NP
      TIME = MAX(TIME, TT6(I) - TT5(I))
      TIMENL = MAX(TIMENL, TT2(I) - TT1(I))
      OPTIME = MAX(OPTIME,TT6(I) - TT2(I))
      WRITE(IW,*)'              Processor # ',I
      WRITE(IW,*)' Nonlinear Analysis Total Time = ',TT6(I) - TT5(I)
      WRITE(IW,*)' Optimization Total Time       = ',TT2(I) - TT1(I)
      WRITE(IW,*)' TOTAL TIME                    = ',TT6(I) - TT2(I)
202   CONTINUE
      WRITE(IW,*)
      WRITE(IW,*)' ************************************************'
      WRITE(IW,*)' Total Number of M.Operation Count = ',NOOCT/1.0E06
      WRITE(IW,*)' ************************************************'
      WRITE(IW,*)' Nonlinear Analysis Total Time = ',TIMENL
      WRITE(IW,*)' Optimization Total Time       = ',OPTIME
      WRITE(IW,*)' TOTAL TIME                    = ',TIME
      FLOPNL = NOOCT / (TIMENL*1000000)
      WRITE(IW,*)' Nonlinear M.Flop Rate         = ',FLOPNL
      End Barrier
      Barrier
      End barrier
115   FORMAT(//2X,'NUMBER OF NODES =',I6
     & /2X,'NUMBER OF ELEMENTS TYPE =',I6//)
125   FORMAT(//15X,'********* T R U S S  E L E M E N T **********'
     & //2X,'NUMBER OF ELEMENT =',I6
     & /2X,'NUMBER OF MATERIAL PROPERTY =',I6//)
135   FORMAT(//15X,'********* B E A M  E L E M E N T **********'
     & //2X,'NUMBER OF ELEMENT =',I6
     & /2X,'NUMBER OF MATERIAL PROPERTY =',I6//)
1000  FORMAT(A72)
1001  FORMAT(10I5)
1002  FORMAT(9I5)
1009  FORMAT(I5,I5,I5,I5)
1012  FORMAT(3F10.3)
      Join
      END
C **.*****************************************************************
C  This subroutine AAAA will save the global stiffness matrix into   *
C  another array in order to be used with P.C.G. as a preconditioner *
C **.*****************************************************************
      SUBROUTINE AAAA(A,AA,NTRMS)
      REAL A(1),AA(1)
      DO 10 I = 1 , NTRMS
      AA(I) = A(I)
10    CONTINUE
```

```
      RETURN
      END
C **.*****************************************************************
C  This subroutine ARCLM will find the maximum load for the          *
C  nonlinear problem if other methods fails.                         *
C **.*****************************************************************
      SUBROUTINE ARCLM (IR,IW,NOMP,NELMNT,NNPE,YM,CTH,DM,AR,WIGHT,
     & NODELM,AL,CX,CY,CZ,X,Y,Z,MPN,ICHK,ALO,NUM4,MULT4,MID,AXF,ID,
     & NNODE,NDPN ,NTD,DELP ,P,NEWIT,DELD,TOTLD,MESTOP,CNORM,FNORM,
     & DELMDA )
      REAL YM(1) , AR(1) ,CX(1),CY(1),CZ(1) , AL(1),ALO(1),AXF(1)
      REAL X(1) , Y(1) , Z(1) ,CTH(1),DM(1),WIGHT(1),DELP(1),P(1)
      REAL DELD(1),TOTLD(1),DD(6)
      INTEGER ID(6,NNODE)
      INTEGER NODELM(NNPE,NELMNT),MPN(1),MID(1)
      IF (MESTOP .EQ. 1 ) GO TO 19
      SUM = 0.0
      DO 10 I = 1 , NTD
      TEMP =  DELD(I)
      SUM = SUM + TEMP**2
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW,*)'  I = ',I,'   DELD(I) =',DELD(I)
      ENDIF
      DELD(I) = 0.0
10    CONTINUE
      CNORM = SUM
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW,*)' DELTA L ** 2 = ',SUM
      ENDIF
      GO TO 29
19    CONTINUE
      WRITE(IW,*)'  C NORM ',CNORM
      WRITE(IW,*)'  F NORM ',FNORM
      DELMDA = SQRT ( CNORM / FNORM  )
      WRITE(IW,*)'  DELTA LAMDA = ',DELMDA
      DO 20 I = 1 , NTD
      TOTLD(I) = TOTLD(I) - DELD(I)
      WRITE(IW,*)'  I = ',I,'   TOTLD(I) =',TOTLD(I)
20    CONTINUE
      DO 40 I = 1 , NNODE
      DO 30 J = 1 , 6
      IF(ID(J,I) .NE. 0 ) THEN
      ICONT = ICONT + 1
      DD(J)  = DELD(ICONT)
      ELSE
      DD(J) = 0.0
      ENDIF
30    CONTINUE
      X(I) = X(I) - DD(1)
      Y(I) = Y(I) - DD(2)
      Z(I) = Z(I) - DD(3)
      WRITE(IW,25)I,(DD(J),J=1,6)
      WRITE(IW,*)
      WRITE(IW,*)
40    CONTINUE
      IF(ICHK .EQ. 1 ) THEN
      WRITE(IW,35)
      DO 31 I = 1 , NNODE
      WRITE(IW,45) I,X(I),Y(I),Z(I)
31    CONTINUE
      CALL       NEWCTS (IR,IW,NOMP,NELMNT,NNPE,YM,CTH,DM,AR,WIGHT,
     & NODELM,AL,CX,CY,CZ,X,Y,Z,MPN,ICHK,ALO,NUM4,MULT4,MID,AXF,ID,
     & NNODE,NDPN ,NTD,DELP ,P,NEWIT,DELMDA,EPS1,LSTOP)
      ENDIF
29    CONTINUE
25    FORMAT(/2X,I5,/3E16.6/7x,3E16.6)
35    FORMAT(//10X,' N E W   C O O R D I N A T E S '// )
45    FORMAT(/2X,I6,3F15.5 )
      RETURN
      END
C **.*****************************************************************
C  This subroutine AXILA will find the axial forces and stresses in  *
C  linear analysis for truss element.                                *
C **.*****************************************************************
      SUBROUTINE AXIAL ( IR,IW,NNODE,NELMNT,NNPE, D,ID,NODELM,ICHK,
     & NDPN,CX,CY,CZ,YM,AR,MID,AL,AXF,DLTP,NTD)
      REAL D(1),DD(6),CX(1),CY(1),CZ(1),YM(1),AR(1),AL(1),AXF(1)
      REAL DLTP(1)
      INTEGER ID(6,NNODE), LM(24) ,MID(1) ,NODELM(NNPE,NELMNT)
      WRITE(IW,15)
      DO 20 I = 1 , NELMNT
      CALL       SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,I,
     & NDPN )
      DO 10 J = 1 , 6
      ICONST = LM(J)
      IF(ICONST .EQ. 0 )THEN
      DD(J) = 0
      ELSE
```

```
      DD(J) = D(ICONST)
      ENDIF
10    CONTINUE
      E = YM ( MID(I) )
      A = AR ( MID(I) )
      AXFRCE = ( ( DD(4)-DD(1) ) * CX(I) + ( DD(5)-DD(2) ) * CY(I) +
     & ( DD(6) - DD(3) ) * CZ(I) ) * ( (A *  E) /AL(I) )
      AXF(I) = AXFRCE
      WRITE(IW,55) I , AXFRCE
20    CONTINUE
25    FORMAT(I10 ,F20.4)
55    FORMAT(/2X, I6 , E16.6 )
15    FORMAT(20X,' M E M B E R   F O R C E S '
     & /19X,    '=============================' )
      RETURN
      END
C **.*****************************************************************
C   This subroutine BEAMST will find element end forces for the BEAM  *
C   elemet in linear analysis.                                        *
C **.*****************************************************************
      SUBROUTINE BEAMST
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM ,PR ,  DM,WEIGHT,D,
     & A1 ,  A2   , A3  ,  AI1 ,  AI2 , AI3 ,
     & AL   ,C ,BIGK ,ID    ,NODELM ,MID,MEID,NID,NJPROW,MAXA)
      REAL  X(1),Y(1),Z(1),YM(1),PR(1),DM(1),WEIGHT(1)
      REAL  A1(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL  BIGK(1),D(1)
      REAL  AL(1),C(9,NELMNT),SMLK(12,12),DD(12),D3(12),P(4000)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24),MEID(1)
      do 300 ii = 1 , 4000
      p(ii) = 0.0
300   continue
      WRITE(IW,*)'            M E M B E R   E N D   F O R C E S '
      WRITE(IW,*)
      DO 20 I = 1 , NELMNT
      CALL      SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,I,
     & NDPN )
      DO 10 J = 1 , 12
      ICONST = LM(J)
      IF(ICONST .EQ. 0 )THEN
      DD(J) = 0
      ELSE
      DD(J) = D(ICONST)
      ENDIF
10    CONTINUE
      E =    YM( MID( I ) )
      V =    PR(MID(I))
      G =    E / (2.0 * ( 1.0 + V ) )
      AREA = A1( MEID( I ) )
      AIX  = AI1 ( MEID ( I ) )
      AIY  = AI2 ( MEID ( I ) )
      AIZ  = AI3 ( MEID ( I ) )
      ALL  = AL(I)
      CALL  ES3DB0(I,NELMNT,NNODE,SMLK,AREA,E,AIX,AIY,AIZ,C,ALL,
     & NTD,G,M1,N1,ICHK,DD,D3 )
      DO 40 II = 1 , 6
      JJ = LM(II)
      MM = LM(II+6)
      IF(JJ . NE. 0 ) THEN
      P(JJ) = P(JJ) +  D3(II)
      ENDIF
      IF( MM .NE. 0 ) THEN
      P(MM) = P(MM) + D3(II+6)
      ENDIF
40    CONTINUE
      UFNORM = 0.0
      DO 60 II = 1 , NTD
      UFNORM = UFNORM + P(II) * P(II)
      IF(UFNORM .GT. 1.0E20) GO TO 91
60    CONTINUE
91    UFNORM = SQRT(UFNORM)
      WRITE(IW,*)' UNBALANCED FORCE NORM = ',UFNORM
      ENDIF
20    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine BFGS contains subroutine BFGS1 which includes the *
C   BFGS method.                                                      *
C **.*****************************************************************
      Forcesub BFGS (A,KA,W,NOOCT) of NP ident ME
      REAL A(1)
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
```

```
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
     COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      Shared COMMON/BF1/MULT4,NUM4
      Shared REAL T1(16),T2(16),T3(16),T4(16),T5(16),T6(16),T7(16)
      Shared REAL TOL,EPS,TIMELT(16),TIMEAM(16),TIMEBS(16)
      Shared REAL TTNOR,TDISP(16),TNEW(16)
      Shared INTEGER JSTOP,MESTOP,LSTOP,NUM4S
      Shared INTEGER NOOCA,NOOCB,NOOCC,NOOCU,NOOCD,NOOCZ
      Shared LOGICAL  TYPE1
      Private INTEGER NUM4P,NADYA,MARY
      Externf TRUSS,ROWC,DISPF,UBFITS,BFGS1,PBEAM,UBFIB2
      End declarations
      ISTOP = 2
      CALL CPUTIM(T6(ME))
      TIMELT(ME)  = 0.0
      TIMEBS(ME)  = 0.0
      TIMEAM(ME)  = 0.0
      TDISP(ME) = 0.0
      TNEW(ME) = 0.0
      NOOCA = 0
      NOOCB = 0
      NOOCC = 0
      NOOCD = 0
      NOOCU = 0
      NOOCZ = 0
      TOL = EPS1
      EPS = EPS1
      MESTOP = 0
      DO 20 NUM4P = 1 , NLD
      ISTOP = 2
      Barrier
      NUM4 = NUM4P
      End barrier
      Critical TYPE1
      JSTOP = 0
      LSTOP = 0
      TTNOR = 0.0
      End critical
      DO 19 MARY = 1 ,10
      Barrier
      End Barrier
      Critical TYPE1
      JSTOP = 0
      LSTOP = 0
      TTNOR = 0.0
      WRITE(6,*)' ME = ',ME
      End critical
      DO 90 NADYA = 1 , ISTOP
      Barrier
      End Barrier
      Barrier
      CALL CPUTIM(T1(ME))
      WRITE(IW,95) NUM4
      WRITE(IW,195) NADYA
      MULT4 = NUM4 - 1
      DO 101 ITRQ = K20 , KL20 -1
      A(ITRQ) = 0.0
101   CONTINUE
      CALL CPUTIM(T2(ME))
      End Barrier
      Barrier
      End barrier
      CALL CPUTIM(T4(ME))
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall TRUSS
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
     & A(K10).A(K11).A(K12),A(K13),A(K20),KA(M1),KA(MI),
     & KA(M2),KA(M3),KA(M3A),KA(M4),KA(M4A),KA(M6),A(K16),NOOCA)
      ELSEIF ( JFLAG .EQ. 2 ) THEN
      Forcecall PBEAM
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
```

```
& KA(M2),KA(M2A),KA(M3),KA(M3A),KA(M4),KA(M4A), KA(M6),A(K16),
& NOOCA)
 ENDIF
 CALL CPUTIM(T5(ME))
 NEQP1 = NTD + 1
 CALL CPUTIM(T1(ME))
 JOPS = 0
 Forcecall ROWC
&       (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
& NTRMS,1,JOPS,ICHK,NUM4,MESTOP)
 CALL CPUTIM(T2(ME))
 Barrier
 IF(MESTOP .EQ. 1 ) THEN
 WRITE(IW,*) '*********************************************'
 WRITE(IW,*) '* S W I T C H  T O  A R C  L E N G T H *'
 WRITE(IW,*) '*********************************************'
 ENDIF
 End barrier
 IF(MESTOP .EQ. 1 ) STOP
 JOPS1 = 0
 Forcecall ROWC
&       (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
& NTRMS,2,JOPS1,ICHK,NUM4,MESTOP)
 CALL CPUTIM(T3(ME))
 Critical TYPE1
 TIMELT(ME) = T2(ME) - T1(ME) + TIMELT(ME)
 TIMEBS(ME) = T3(ME) - T2(ME) + TIMEBS(ME)
 TIMEAM(ME) = T5(ME) - T4(ME) + TIMEAM(ME)
 NOOCC = NOOCC + JOPS
 NOOCB = NOOCB + JOPS1
 End critical
 CALL CPUTIM(T1(ME))
 IF(NADYA .GE. ISTOP   ) THEN
 Forcecall BFGS1(A(1),KA(1),A(K14),A(K14A),A(K15),A(K40),A(K41),
& A(K42),NOOCA,NOOCb,NOOCC,NOOCU,NOOCD,NOOCZ,ISTOP)
 ENDIF
 CALL CPUTIM(T2(ME))
 Critical TYPE1
 JSTOP = 0
 TTIME = T2(ME) - T1(ME)
 WRITE(6,*)' ME = ',ME,'    TIME BFGS1 = ',TTIME
 End critical
 Barrier
 End barrier
 CALL CPUTIM(T2(ME))
 IF(NADYA .LT. ISTOP ) THEN
 Forcecall DISPF (IR,IW,NNODE ,A(K14),A(K14A),KA(M1),A(K15),NTD,
& A(K1),A(K2),A(K3),ICHK,NUM4,ISWTCH,NOOCD)
 ENDIF
 CALL CPUTIM(T1(ME))
 TDISP(ME) = TDISP(ME) + T1(ME) - T2(ME)
 Barrier
 End barrier
 CALL CPUTIM(T1(ME))
 IF( JFLAG .EQ. 1 ) THEN
 Forcecall UBFITS (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(K6),A(K7),
& A(K8),KA(MI),A(K10),A(K11),A(K12),A(K13),A(K1),A(K2),A(K3),
& KA(M2) ,ICHK ,A(K10A),NUM4,MULT4 ,KA(M2) ,A(K16) ,KA(M1),
& NNODE,NDPN,NTD,A(K17),A(K14),NADYA,DELMDA,EPS1,METHOD,JSTOP,
& NOOCU )
 ELSEIF (JFLAG .EQ. 2 )THEN
 Forcecall UBFIB2
&       (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
& NTRMS,NOMP,NNPE,ICHK,MULT4,
& A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K15),
& A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
& A(K10A) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
& KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),A(K14),
& A(K17),TOL,JSTOP,METHOD,NUM4 ,A(K10),NOOCU)
 ENDIF
 Barrier
 CALL     WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
& ICHK)
 End Barrier
 CALL CPUTIM(T2(ME))
 TNEW(ME) = TNEW(ME) + T2(ME) - T1(ME)
 Barrier
 WRITE(IW,*) ' JSTOP =' ,JSTOP
 End barrier
 IF(JSTOP .EQ. 1 ) GO TO 20
90  CONTINUE
19  CONTINUE
20  CONTINUE
 Barrier
 TMAXLT = 0.0
 TMAXBS = 0.0
 TMAXAM = 0.0
 TMAXUF = 0.0
```

```
      TMAXDP = 0.0
      WRITE(IW,*)
      WRITE(IW,*) ' N U M B E R   O F   P R O C E S E S = ', NP
      WRITE(IW,*)
      WRITE(IW,*)'*******************************************'
      WRITE(IW,*)' Assembly Operation Count      = ',NOOCA/1.0E06
      WRITE(IW,*)' Factorization Operation Count = ',NOOCC/1.0E06
      WRITE(IW,*)' B. Substitution Operation Count = ',NOOCB/1.0E06
      WRITE(IW,*)' U.B.Forces      Operation Count = ',NOOCU/1.0E06
      WRITE(IW,*)' Displacement    Operation Count = ',NOOCD/1.0E06
      WRITE(IW,*)'*******************************************'
      CALL      WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
     & ICHK)
      DO 100 I = 1 , NP
      TMAXLT = MAX(TMAXLT , TIMELT(I) )
      TMAXBS = MAX(TMAXBS , TIMEBS (I))
      TMAXAM = MAX(TMAXAM , TIMEAM(I) )
      TMAXUF = MAX(TMAXUF , TNEW  (I) )
      TMAXDP = MAX(TMAXDP , TDISP (I) )
      WRITE(IW,*)' Process No. = ',I,' ASSEMBLE    TIME = ',TIMEAM(I)
      WRITE(IW,*)' Process No. = ',I,' L.T.        TIME = ',TIMELT(I)
      WRITE(IW,*)' Process No. = ',I,' B.S.        TIME = ',TIMEBS(I)
      WRITE(IW,*)' Process No. = ',I,' U.B. FORCES TIME = ',TNEW  (I)
      WRITE(IW,*)' Process No. = ',I,' DISP        TIME = ',TDISP (I)
      WRITE(IW,*)
  100  CONTINUE
      WRITE(IW,*) '================================='
      WRITE(IW,*) '=======     Max Time Calculations    ==========='
      WRITE(IW,*) '================================='
      WRITE(IW,*)
      WRITE(IW,*) ' ASSMEBLE    MAX TIME    = ',TMAXAM
      WRITE(IW,*) ' FACTORIZE   MAX TIME    = ',TMAXLT
      WRITE(IW,*) ' B.S.        MAX TIME    = ',TMAXBS
      WRITE(IW,*) ' U.B. FORCES MAX TIME    = ',TMAXUF
      WRITE(IW,*) ' DISP        MAX TIME    = ',TMAXDP
      WRITE(IW,*) '================================='
      WRITE(IW,*) '=======  Mega Flop Rate Calculations  =========='
      WRITE(IW,*) '================================='
      WRITE(IW,*)
      WRITE(IW,*) ' ASSMEBLE    FLOP RATE   = ',NOOCA /(TMAXAM*1.0E06)
      WRITE(IW,*) ' FACTORIZE   FLOP RATE   = ',NOOCC /(TMAXLT*1.0E06)
      WRITE(IW,*) ' B.S.        FLOP RATE   = ',NOOCB /(TMAXBS*1.0E06)
      WRITE(IW,*) ' U.B. FORCES FLOP RATE   = ',NOOCU/(TMAXUF *1.0E06)
      WRITE(IW,*) ' DISP        FLOP RATE   = ',NOOCD/(TMAXDP *1.0e06)
      WRITE(IW,*)'================================='
      NOOCT = NOOCA + NOOCB + NOOCC + NOOCU + NOOCD+NOOCZ
      WRITE(IW,*) ' ASSMEBLE    MAX TIME    = ',TMAXAM
      WRITE(IW,*) ' L.T.        MAX TIME    = ',TMAXLT
      WRITE(IW,*) ' B.S.        MAX TIME    = ',TMAXBS
      WRITE(IW,*) ' U.B. FORCES MAX TIME    = ',TMAXUF
      WRITE(IW,*) ' DISP        MAX TIME    = ',TMAXDP
      WRITE(IW,*)' TIME FOR DISP     = ',TDISP
      WRITE(IW,*)' TIME FOR TNORM    = ',TTNOR
      WRITE(IW,*)' TIME FOR NEWCTS   = ',TNEW
      IF( NP .EQ. 1) THEN
      WRITE(8) TMAXLT,TMAXBS,TMAXAM
      ELSE
      REWIND(8)
      READ(8) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR ASMBLE     = ',RMAXAM /TMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR L.T.       = ',RMAXLT / TMAXLT
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR B.S        = ',RMAXBS / TMAXBS
      WRITE(IW,*)
      ENDIF
      End Barrier
      CALL CPUTIM(T7(ME))
      Critical TYPE1
      TNEWTN = T7(ME) -T6(ME)
      WRITE(IW,*) ' ME = ', ME,'   TOTAL TIME IN BFGS   = ',TNEWTN
      End critical
      Barrier
      End barrier
   95   FORMAT(15X,'********** I T E R A T I O N   N O : ',I5)
  195   FORMAT(15X,'.......... N E W T O N       N O : ',I5)
  999   RETURN
      END
C **.*********************************************************
C   This subroutine BFGS1 is the BFGS method algorithm.       *
C **.*********************************************************
      Forcesub BFGS1 (A,KA,D,DLD,U,W,V,F,
     & NOOCA,NOOCB,NOOCC,NOOCU,NOOCD,NOOCT,ISTOP)
     & of NP ident ME
      REAL A(1)
```

```
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
     COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      COMMON/BF1/MULT4,NUM4
      REAL D(1),DLD(1),U(1),W(NTD,NOIIT),V(NTD,NOIIT),F(1)
      Shared REAL T1(16),T2(16),T3(16),T4(16),T5(16),T6(16),T7(16)
      Shared REAL TOL,EPS,TIMELT(16),TIMEAM(16),TIMEBS(16)
      Shared REAL TTNOR,TDISP(16),TNEW(16)
      Shared REAL DTEMP(9000),S,SUM1,SUM2,SUM3,CON1,CON2,CONTOL,SUM4
      Shared REAL HNORM,SUM5,CHEK,TT1(16),TALFA,TT2(16),THNORM,OSTOPP
      Shared INTEGER JSTOP,MESTOP,LSTOP
      Private INTEGER MUNA,J,ICONT
      Private REAL SUM1P,SUM2P,SUM3P,CON1P,CON2P,PHNORM,SUM4P
      Shared LOGICAL  TYPE4,TYPE5,TYPE6
      Externf UBFITS,ROWC,DISPF,ROOT,UBFIB2
      End declarations
      do 1000 i = 1 , ntd
      dtemp(I) = U(I)
1000  continue
      THNORM = 1.0E49
      TOL=EPS1
      EPS = EPS1
      TALFA = 0.0
      N = NTD
      Barrier
      WRITE(6,95) NOIIT
      HNORM = 1.0E49
      End barrier
      S=1.00
      ICONT=0
      rewind (12)
      Presched do 20 I=1,N
      read(12)F(I)
      DO 10 J=1,NOIIT
      W(I,J)=0.0
      V(I,J)=0.0
10    CONTINUE
20    End presched do
C ---------- PROGRAM START ----------
      DO  110 J=1,NOIIT
      Barrier
      End barrier
      Presched do 40 I=1,N
      V(I,J)=F(I)
40    End presched do
      Barrier
      End barrier
      CALL CPUTIM(TT1(ME))
      Forcecall ROOT (A,KA,S,NOOCU)
      CALL CPUTIM(TT2(ME))
      Barrier
      HNORML = HNORM
      End barrier
      ICONT=ICONT+1
      Presched do 41 I = 1 , N
      DLD(I) = S * D(I)
      W(I,J) = U(I)
41    End presched do
      Barrier
      NOOCT = NOOCT + N
      TALFA = TALFA + TT2(ME) - TT1(ME)
      End barrier
      Forcecall DISPF (IR,IW,NNODE ,DLD,DTEMP,KA(M1),U,NTD,
     & A(K1),A(K2),A(K3),ICHK,NUM4,ISWTCH,NOOCD)
      Barrier
      End barrier
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall UBFITS (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(K6),A(K7),
     & A(K8),KA(M1),A(K10),A(K11),A(K12),A(K13),A(K1),A(K2),A(K3),
     & KA(M2) ,ICHK ,A(K10A),NUM4,MULT4 ,KA(M2) ,A(K16) ,KA(M1),
     & NNODE,NDPN,NTD,A(K17),F,NEWIT,DELMDA,EPS1,1,LSTOP,NOOCU)
      ELSEIF ( JFLAG .EQ. 2 ) THEN
      Forcecall UBFIB2
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MULT4,
```

```
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),U,
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10A) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),F,
     & A(K17),TOL,JSTOP,-3,NUM4 ,A(K10),NOOCU )
       ENDIF
       Barrier
       End barrier
       HNORM = 0.0
       SUM1=0.0
       SUM2=0.0
       SUM3=0.0
       SUM1P=0.0
       SUM2P=0.0
       SUM3P=0.0
       PHNORM = 0.0
CDIR$ IVDEP
       Presched do 51 I= 1    , N
       CON1P= - 1.0 * (F(I)-V(I,J))
       CON2P = DLD(I)
       SUM1P=SUM1P+(D(I)*CON1P)
       SUM2P=SUM2P+(V(I,J)*CON2P)
       SUM3P=SUM3P+(CON2P*CON1P)
       PHNORM = PHNORM + F(I) * F(I)
51     End presched do
       Barrier
       NOOCT = NOOCT + 10 * N
       End barrier
       Critical TYPE5
       SUM1 = SUM1 + SUM1P
       SUM2 = SUM2 + SUM2P
       SUM3 = SUM3 + SUM3P
       HNORM = HNORM + PHNORM
       End critical
       Barrier
       End barrier
       Barrier
       HNORM = SQRT(HNORM)
       OSTOPP = THNORM / HNORM
       THNORM = HNORM
       End barrier
       IF(HNORML .LT. HNORM ) THEN
       ISTOP = 9
       ENDIF
       IF(HNORM.LT.TOL ) GO TO 120
       IF(HNORML .LT. HNORM ) GO TO 120
       IF(SUM2 .EQ. 0.0) GO TO 120
       Barrier
       CHEK=SUM1/SUM2
       End barrier
       IF(CHEK.LT.0.0) GO TO 120
       Barrier
       CONTOL=-SQRT(SUM1/SUM2)*S
       CONTOL=CONTOL+1.0
       End barrier
       SUM4 = 0.0
       SUM4P = 0.0
       Presched do 60 I=1,N
       W(I,J)=(1.0/SUM3)* DLD(I)
       SUM4P = SUM4P + ( W(I,J) * F(I) )
60     End presched do
       Barrier
       NOOCT = NOOCT + 9 * N
       End barrier
       Critical TYPE6
       SUM4 = SUM4 + SUM4P
       End critical
       Barrier
       End barrier
       Presched do 57 I = 1 , N
       V(I,J)=CONTOL*V(I,J)+F(I)
       D(I) = F(I) + ( SUM4 * V(I,J) )
57     End presched do
C ..... T H E   J   L O O P   S T A R T S .....
       Barrier
       DO 74 L=J-1,1,-1
       SUM6=0.0
       DO 72 I=1,N
72     SUM6=SUM6+(W(I,L)*D(I))
       DO 73 I=1,N
73     D(I)=D(I)+(SUM6*V(I,L))
       NOOCT = NOOCT + 4 * N
74     CONTINUE
       End barrier
       NEQP1 = N + 1
       JOPS = 0
       Forcecall ROWC
     &       (A(K20),D,KA(M6),KA(M7),KA(M5),N  ,NEQP1,
```

```
     & NTRMS,2,JOPS,ICHK,NUM4,MESTOP)
       Critical TYPE4
       NOOCB = NOOCB + JOPS
       End critical
       DO 81 L=1,J
       Barrier
 61    SUM5=0.0
       DO 90 I=1,N
 90    SUM5=SUM5+V(I,L)*D(I)
       NOOCT = NOOCT + 4 * N
       End barrier
       IF( HNORM   .LT. TOL) GO TO 120
       Barrier
       DO 100 I=1,N
       D(I)=D(I)+W(I,L)*SUM5
 100   CONTINUE
       End barrier
 81    CONTINUE
       Barrier
       End barrier
 110   CONTINUE
 300   CONTINUE
 120   CONTINUE
       Barrier
       WRITE(6,45) ICONT
       WRITE(IW,*)' TOTAL TIME FOR ALFA = ',TALFA
       End barrier
 15    FORMAT (/F20.8,/F20.8/F20.8)
 25    FORMAT(/3F20.2/3F20.2/)
 35    FORMAT(4F17.8)
 45    FORMAT(/2X,'NUMBER OF ITERATIONS =',I8//,2X,' THE SOLUTION IS'/)
 85    FORMAT(2X,'TOL= ',E10.2)
 95    FORMAT(20X,'**********************************',/
     & 20X,      '*  NUMBER OF STORAGE = ',I5,'  *',/
     & 20X,      '**********************************' )
       RETURN
       END
C **.*****************************************************************
C  This subroutine COLH will find the column hight of the global      *
C  tangent stiffness matrix.                                          *
C **.*****************************************************************
       SUBROUTINE COLH (IR,IW,NNODE,NELMNT,NTD,ID,ICOLH ,MAXA,NTRMS,
     & NDPE,NODELM,NNPE,ICHK,NDPN )
       INTEGER ICOLH(1),ID(6,NNODE),MAXA(1),NODELM(NNPE,NELMNT)
       INTEGER LM(24)
       ICOLH(1) = 0
       DO 102 I = 1 , NTD
102    ICOLH(I) = 0
       DO 1 I = 1 , NELMNT
       CALL SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,I,
     & NDPN )
       DO 2 J = 1 , NDPE
       LL = LM(J)
       DO 3 K = 1 , NDPE
       KK = LM(K)
       ITEMP = LL - KK
       IF(LL.EQ.0. OR. KK.EQ.0 ) GO TO 3
       IF(ITEMP .GT. ICOLH(LL))  ICOLH(LL) = ITEMP
 3     CONTINUE
 2     CONTINUE
 1     CONTINUE
       MAXA(1) = 1
       DO 50 I = 2 , NTD + 1
       MAXA(I) = ICOLH(I-1) + MAXA(I-1) + 1
 50    CONTINUE
       NTRMS = MAXA(NTD) + ICOLH(NTD)
       RETURN
       END
C **.*****************************************************************
C  This subroutine CPUTIM will find the time spend in the analysis.   *
C  TSECND() = CPU time on CRAY.                                       *
C  TIMEF()  = Wall clock time. (CRAY)                                 *
C  TSECND() = Total time by all processors. (CRAY)                    *
C **.*****************************************************************
       SUBROUTINE CPUTIM(ITIME)
       REAL ITIME
       ITIME = TSECND()
C      ITIME = SECOND()
C      ITIME = TIMEF()
       RETURN
       END
C **.*****************************************************************
C  This subroutine DIAMTR will find the outside and inside diameter   *
C  for any pipe section provided the area and the moments of ineria.  *
C **.*****************************************************************
       SUBROUTINE DIAMTR
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NOEP,NNPE,ICHK,IOPT,INOMP,DISPA,SIGMAA,
```

```
     & A1 ,  A2   , A3  ,  AI1 ,  AI2 , AI3,DIN,DOUT )
      REAL A1(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL DIN(1),DOUT(1)
      PI = ACOS(-1.0)
      WRITE(6,*)' PI = ',PI
      WRITE(IW,*) ' Number of Elemnt Property ',NOEP
      DO 10 I = 1  , NOEP
      A = ( 8.0 * A1(I) / PI )
      B = 0.0
      C = (16.0 * (A1(I)**2) / (PI**2) ) - ( 32.0 * AI1(I) / PI )
      DIND    = ( -B + SQRT( B**2 - 4.0 * A * C ) ) / ( 2.0 * A )
      DOUTT   = SQRT ( (DIND**2) + ( 4.0 * A1(I) / PI ) )
      DIN(I)  = DIND
      DOUT(I) = DOUTT
      IF(NOEP .LT. 4 ) THEN
      WRITE(IW,*)
      WRITE(IW,*)'  Inside  Diam.  ( ',I,') = ',DIN(I)
      WRITE(IW,*)'  Outside Diam.  ( ',I,') = ',DOUT(I)
      ENDIF
10    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine DISPF will be used to find the new position of    *
C   the nodes.                                                        *
C **.*****************************************************************
      Forcesub DISPF (IR,IW,NNODE ,D,DLD,ID,TOTLD,NTD,X,Y,Z,ICHK,
     & NUM4,ISWTCH,NOOC)
     & of NP ident ME
      REAL D(1),TOTLD(1),X(1),Y(1),Z(1),DLD(1)
      INTEGER ID(6,NNODE)
      Private INTEGER ICONT
      Private REAL DD(6)
      Shared LOGICAL TYPE1
      End declarations
      ICONT = 0
      NOOCP = 0
      Presched do 20 I = 1 , NNODE
      DO 10 J = 1 , 6
      ICONT = ID(J,I)
      IF(ICONT .NE. 0 ) THEN
      DD(J)  = D(ICONT)
      NOOCP = NOOCP + 1
      TOTLD(ICONT) = TOTLD(ICONT) + D(ICONT)
      DLD(ICONT) = D(ICONT)
      ELSE
      DD(J) = 0.0
      ENDIF
10    CONTINUE
      X(I) = X(I) + DD(1)
      Y(I) = Y(I) + DD(2)
      Z(I) = Z(I) + DD(3)
      NOOCP = NOOCP + 3
20    End presched do
      Critical TYPE1
      NOOC = NOOC + NOOCP
      End critical
      RETURN
      END
C **.*****************************************************************
C   This subroutine ES3DB1 will find the the Kii  and Kij of the      *
C   element stiffness matrix. (Beam element)                          *
C **.*****************************************************************
      SUBROUTINE ES3DB1(J,NE,NJ,K,AX,E,IX,IY,IZ,C,L,NOTDOF,G,M1,N1,ICHK,
     & BB)
      REAL  K1(6,6),K2(6,6),D1(6,6),D2(6,6),T(6,6),K(12,12)
      REAL  C(9,NE),L,IX,IY,IZ
      IW = 6
      DO 10 NY =  1 ,  6
      DO 10 MY =  1 ,  6
      K1(NY,MY) = 0.0
      K2(NY,MY) = 0.0
10    CONTINUE
      M2=M1+1
      M3=M1+2
      M4 = M1 + 3
      M5 = M1 + 4
      M6 = M1 + 5
      N2 = N1 + 1
      N3 = N1 + 2
      N4 = N1 + 3
      N5 = N1 + 4
      N6 = N1 + 5
      U1 = AX * E / L
      U2 = 12.0 * E / ( L**3 )
      U3 = G / L
      U4 = 4 * E / L
      U5 = 6.0 * E / ( L**2 )
```

```
      K1(M1,M1) = U1
      K1(M2,M2) = U2 * IZ + BB
      K1(M2,M6) = U5 * IZ
      K1(M3,M3) = U2 * IY + BB
      K1(M3,M5) = - U5 * IY
      K1(M4,M4) = U3 * IX
      K1(M5,M3) = K1(M3,M5)
      K1(M5,M5) = U4 * IY
      K1(M6,M2) = K1(M2,M6)
      K1(M6,M6) = U4 * IZ
      DO  20 NN = 1 , 6
      DO  20 MM = 1 , 6
      K2(NN , MM ) =  - K1(NN,MM)
20    CONTINUE
      K2(M3,M5) = K1(M3,M5)
      K2(M2,M6) = K1(M2,M6)
      K2(M5,M5) = K1(M5,M5) / 2.0
      K2(M6,M6) = K1(M6,M6) / 2.0
      CALL      TRANS ( NE , T , C , J ,ICHK )
      CALL      MATM  ( 6 , 6 , 6 ,K1,T,D1)
      CALL      MATM  ( 6 , 6 , 6 ,K2,T,D2)
      CALL      TRANST ( NE ,T , C , J , ICHK )
      CALL      MATM2  ( 6 , 6 , 6 ,T,D1,K1)
      CALL      MATM2  ( 6 , 6 , 6 ,T,D2,K2)
      DO 30 NN = 1 , 6
      DO 30 MM = 1 , 6
      K(NN,MM) = K1(NN,MM)
30    CONTINUE
      DO 40 NN = 1 , 6
      NCONT = 1
      DO 40 MM = 7 , 12
      K(NN,MM) = K2(NN,NCONT)
      NCONT = NCONT + 1
40    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine ES3DB1 will find the the Kjj of the element      *
C   stiffness matrix. (Beam element)                                 *
C **.*****************************************************************
      SUBROUTINE ES3DB2(J,NE,NJ,K,AX,E,IX,IY,IZ,C,L,NOTDOF,G,M1,N1,ICHK,
     & BB )
      REAL  K1(6,6),D1(6,6),T(6,6),K(12,12)
      REAL  C(9,NE),L,IX,IY,IZ
      IW = 6
      DO 10 NY =  1 ,  6
      DO 10 MY =  1 ,  6
      K1(NY,MY) = 0.0
10    CONTINUE
      M2=M1+1
      M3=M1+2
      M4 = M1 + 3
      M5 = M1 + 4
      M6 = M1 + 5
      N2 = N1 + 1
      N3 = N1 + 2
      N4 = N1 + 3
      N5 = N1 + 4
      N6 = N1 + 5
      U1 = AX * E / L
      U2 = 12.0 * E / ( L**3 )
      U3 = G / L
      U4 = 4 * E / L
      U5 = 6.0 * E / ( L**2 )
      K1(M1,M1) = U1
      K1(M2,M2) = U2 * IZ + BB
      K1(M2,M6) = - U5 * IZ
      K1(M3,M3) = U2 * IY + BB
      K1(M3,M5) =  U5 * IY
      K1(M4,M4) = U3 * IX
      K1(M5,M3) = K1(M3,M5)
      K1(M5,M5) = U4 * IY
      K1(M6,M2) = K1(M2,M6)
      K1(M6,M6) = U4 * IZ
      CALL      TRANS ( NE , T , C , J , ICHK )
      CALL      MATM  ( 6 , 6 , 6 ,K1,T,D1)
      CALL      TRANST ( NE ,T , C , J ,ICHK )
      CALL      MATM2  ( 6 , 6 , 6 ,T,D1,K1)
      NCONT = 0
      DO 40 NN = 7 , 12
      MCONT = 1
      NCONT = NCONT + 1
      DO 40 MM = 7 , 12
      K(NN,MM) = K1(NCONT,MCONT)
      MCONT = MCONT + 1
40    CONTINUE
      RETURN
      END
```

```
C **.*****************************************************************
C  This subroutine ES3DB1 will find the the total element            *
C  stiffness matrix.  (Beam element)                                 *
C **.*****************************************************************
      SUBROUTINE ES3DBN(J,NE,NJ,AX,E,IX,IY,IZ,C,L,NOTDOF,G,ICHK,
     & D3,D2,AXF,IW )
      REAL R1(12,12),D1(12),D2(12),T(12,12),D3(12),K2(12,12)
      REAL C(9,NE),L,IX,IY,IZ,AXF(1)
      DO 10 NY =  1 ,  12
      DO 10 MY =  1 ,  12
      R1(NY,MY) = 0.0
      T(NY,MY) = 0.0
10    CONTINUE
      M1 = 1
      N1 = 7
      M2=  M1 + 1
      M3=  M1 + 2
      M4 = M1 + 3
      M5 = M1 + 4
      M6 = M1 + 5
      N2 = N1 + 1
      N3 = N1 + 2
      N4 = N1 + 3
      N5 = N1 + 4
      N6 = N1 + 5
      U1 = AX * E / L
      U2 = 12.0 * E / ( L**3 )
      U3 = G / L
      U4 = 4 * E / L
      U5 = 6.0 * E / ( L**2 )
      R1(M1,M1) = U1
      R1(M2,M2) = U2 * IZ
      R1(M2,M6) = U5 * IZ
      R1(M3,M3) = U2 * IY
      R1(M3,M5) = - U5 * IY
      R1(M4,M4) = U3 * IX
      R1(M5,M5) = U4 * IY
      R1(M6,M6) = U4 * IZ
      DO 80 II = 1 , 6
      DO 80 JJ = II , 6
      R1(JJ,II) = R1(II,JJ)
80    CONTINUE
      DO 60 II = 1 , 6
      DO 60 JJ = 7 , 12
      R1(II,JJ) = -R1(II,JJ-6)
60    CONTINUE
      R1(M3,N5) = R1(M3,M5)
      R1(M2,N6) = R1(M2,M6)
      R1(M5,N5) = R1(M5,M5) / 2.0
      R1(M6,N6) = R1(M6,M6) / 2.0
      DO 90 II = 1 , 6
      DO 90 JJ = 7 , 12
      R1(JJ,II) = R1(II,JJ)
90    CONTINUE
      DO 20 II = 7  , 12
      DO 20 JJ = II , 12
      R1(II,JJ) = R1(II-6,JJ-6)
20    CONTINUE
      R1(N2,N6) = - R1(M2,M6)
      R1(N3,N5) = - R1(M3,M5)
      DO 30 II = 7  , 12
      DO 30 JJ = II , 12
      R1(JJ,II) = R1(II,JJ)
30    CONTINUE
      CALL      TRANS1( NE , T , C , J , ICHK,12 )
      CALL      VECMUL ( 12 , 12, 1 ,T ,D2,D1)
      CALL      VECMUL ( 12 , 12, 1 ,R1,D1,D3)
      AXF(J) = - D3(1)
      CALL MATM3 (12,12,12,R1,T,K2)
      CALL TRANS2 (NE,T,C,J,ICHK,12)
      CALL MATM4  (12,12,12,T,K2,R1)
      CALL VECMUL (12,12,1,R1,D2,D3)
      IF( NE .LE. 10 ) THEN
      WRITE(IW,15)J,(D3(II),II=1,12)
      IF(J  .EQ. 1 ) THEN
      WRITE(IW,15)J,(D3(II),II=1,12)
       ELSEIF( J  .EQ. NE/2 ) THEN
       WRITE(IW,15)J,(D3(II),II=1,12)
       ELSEIF( J  .EQ. NE ) THEN
       WRITE(IW,15)J,(D3(II),II=1,12)
       ENDIF
      ENDIF
15    FORMAT(/I10/6E12.5/6E12.5)
45    FORMAT(6E12.3/6E12.3/)
55    FORMAT(3F20.6/3F20.6/3F20.6 )
      RETURN
     END
C **.*****************************************************************
```

```
C  This subroutine ES3DBS will find total elemnt stiffness matrix    *
C  to used in the sensitivity analysis.(Nonlinear Beam element)      *
C **.*****************************************************************
      SUBROUTINE ES3DBS(J,NE,NJ,AX,E,IX,IY,IZ,C,L,NOTDOF,G,ICHK,
     & D3,D2,AXF )
      REAL R1(12,12),D1(12),D2(12),T(12,12),D3(12),K2(12,12)
      REAL C(9,NE),L,IX,IY,IZ,AXF(1)
      DO 10 NY =  1 ,  12
      DO 10 MY =  1 ,  12
      R1(NY,MY) = 0.0
      T(NY,MY) = 0.0
10    CONTINUE
      M1 = 1
      N1 = 7
      M2=  M1 + 1
      M3=  M1 + 2
      M4 = M1 + 3
      M5 = M1 + 4
      M6 = M1 + 5
      N2 = N1 + 1
      N3 = N1 + 2
      N4 = N1 + 3
      N5 = N1 + 4
      N6 = N1 + 5
      U1 = AX * E / L
      U2 = 12.0 * E / ( L**3 )
      U3 = G / L
      U4 = 4 * E / L
      U5 = 6.0 * E / ( L**2 )
      R1(M1,M1) = U1
      R1(M2,M2) = U2 * IZ
      R1(M2,M6) = U5 * IZ
      R1(M3,M3) = U2 * IY
      R1(M3,M5) = - U5 * IY
      R1(M4,M4) = U3 * IX
      R1(M5,M5) = U4 * IY
      R1(M6,M6) = U4 * IZ
      DO 80 II = 1 , 6
      DO 80 JJ = II , 6
      R1(JJ,II) = R1(II,JJ)
80    CONTINUE
      DO 60 II = 1 , 6
      DO 60 JJ = 7 , 12
      R1(II,JJ) = -R1(II,JJ-6)
60    CONTINUE
      R1(M3,N5) = R1(M3,M5)
      R1(M2,N6) = R1(M2,M6)
      R1(M5,N5) = R1(M5,M5) / 2.0
      R1(M6,N6) = R1(M6,M6) / 2.0
      DO 90 II = 1 , 6
      DO 90 JJ = 7 , 12
      R1(JJ,II) = R1(II,JJ)
90    CONTINUE
      DO 20 II = 7  , 12
      DO 20 JJ = II , 12
      R1(II,JJ) = R1(II-6,JJ-6)
20    CONTINUE
      R1(N2,N6) = - R1(M2,M6)
      R1(N3,N5) = - R1(M3,M5)
      DO 30 II = 7  , 12
      DO 30 JJ = II , 12
      R1(JJ,II) = R1(II,JJ)
30    CONTINUE
      CALL      TRANS1( NE , T , C , J , ICHK,12 )
      CALL MATMUL(12,12,12,R1,T,K2)
      CALL TRANS2 (NE,T,C,J,ICHK,12)
      CALL MATMUL (12,12,12,T,K2,R1)
      CALL VECMUL (12,12,1,R1,D2,D3)
      IF( NE .LE. 39 ) THEN
      WRITE(6,15)J,(D3(II),II=1,12)
      ENDIF
15    FORMAT(/I10/6E12.4/6E12.4)
45    FORMAT(6E12.3/6E12.3/)
55    FORMAT(3F20.6/3F20.6/3F20.6 )
      RETURN
      END
C **.*****************************************************************
C  This subroutine ES3dt0 will the total element stiffness matrix    *
C  for the truss element.                                            *
C **.*****************************************************************
      SUBROUTINE ES3DT0(J,NE,NJ,K,A,E,I,CX,CY,CZ,L,NOTDOF,M1,N1)
      REAL  K(6,6)
      REAL      CX(NE),CY(NE),L(NE),I(NE),CZ(NE)
      M2=M1+1
      M3=M1+2
      N2 = N1 + 1
      N3 = N2 + 1
      U1 = A    *E / L(J)
```

```
      K(M1,M1) =  U1 * (CX(J)**2)
      K(M1,M2) =  U1 * CX(J) *CY(J)
      K(M1,M3) =  U1 * CZ(J) *CX(J)
      K(M1,N1) = - K(M1,M1)
      K(M1,N2) = - K(M1,M2)
      K(M1,N3) = - K(M1,M3)
      K(M2,M2) =  U1 * (CY(J)**2)
      K(M2,M3) =  U1 * CY(J) * CZ(J)
      K(M2,N1) = - U1 * CX(J) * CY(J)
      K(M2,N2) = - K(M2,M2)
      K(M2,N3) = - K(M2,M3)
      K(M3,M3) =  U1 * (CZ(J)**2)
      K(M3,N1) =  K(M1,N3)
      K(M3,N2) = - K(M2,M3)
      K(M3,N3) = - K(M3,M3)
      K(N1,N1) =  K(M1,M1)
      K(N1,N2) =  K(M1,M2)
      K(N1,N3) =  K(M1,M3)
      K(N2,N2) =  K(M2,M2)
      K(N2,N3) =  K(M2,M3)
      K(N3,N3) =  K(M3,M3)
      DO 10 IY = 1 , 6
      DO 10 JY = IY , 6
      K(JY,IY) = K(IY,JY)
10    CONTINUE
      RETURN
      END
C **.*********************************************************************
C   This subroutine ES3DT1 will find the contribution of Kii of the   *
C   element stiffness matrix. (Truss element)                         *
C **.*********************************************************************
      SUBROUTINE ES3DT1(J,NE,NJ,K,A,E,I,CX,CY,CZ,L,NOTDOF,M1,N1,BB,
     $ NOOC)
      REAL  K(6,6)
      REAL       CX(NE),CY(NE),L(NE),I(NE),CZ(NE)
      M2=M1+1
      M3=M1+2
      NOOC = NOOC + 17
      U1 =    A *E / L(J)
      K(M1,M1) =  U1 * (CX(J)**2) + BB
      K(M1,M2) =  U1 * CX(J) *CY(J)
      K(M1,M3) =  U1 * CZ(J) *CX(J)
      K(M2,M2) =  U1 * (CY(J)**2) + BB
      K(M2,M3) =  U1 * CY(J) * CZ(J)
      K(M3,M3) =  U1 * (CZ(J)**2) + BB
      DO 10 IY = M1 , N1
      DO 10 JY = IY , N1
      K(JY,IY) = K(IY,JY)
10    CONTINUE
      RETURN
      END
C **.*********************************************************************
C   This subroutine ES3DT2 will find the contribution of Kij of the   *
C   element stiffness matrix. (Truss element)                         *
C **.*********************************************************************
      SUBROUTINE ES3DT2(J,NE,NJ,K,A,E,I,CX,CY,CZ,L,NOTDOF,M1,N1,BB,
     $ NOOC)
      REAL  K(6,6)
      REAL       CX(NE),CY(NE),L(NE),I(NE),CZ(NE)
      M2=M1+1
      M3=M1+2
      N2=N1+1
      N3=N1+2
      NOOC = NOOC + 32
      U1 = A    *E / L(J)
      K(M1,N1) = - U1 * (CX(J)**2) - BB
      K(M1,N2) = - U1 * CY(J) * CX(J)
      K(M1,N3) = - U1 * CZ(J) * CX(J)
      K(M2,N1) = - U1 * CX(J) * CY(J)
      K(M2,N2) = - U1 * (CY(J)**2) - BB
      K(M2,N3) = - U1 * CZ(J) * CY(J)
      K(M3,N1) = - U1 * CX(J) * CZ(J)
      K(M3,N2) = - U1 * CY(J) * CZ(J)
      K(M3,N3) = - U1 * (CZ(J)**2) - BB
      DO 10 IY = M1 , N1
      DO 10 JY = IY , N1
      K(JY,IY) = K(IY,JY)
10    CONTINUE
      RETURN
      END
C **.*********************************************************************
C   This subroutine ES3DT4 will find the contribution of Kij of the   *
C   element stiffness matrix. (Truss element)                         *
C **.*********************************************************************
      SUBROUTINE ES3DT4(J,NE,NJ,K,A,E,I,CX,CY,CZ,L,NOTDOF,M1,N1,BB,
     & NOOC)
      REAL  K(6,6)
      REAL       CX(NE),CY(NE),CZ(NE),L(NE),I(NE)
```

```
      M2=M1+1
      M3=M1+2
      N2=N1+1
      N3=N1+2
      NOOC = NOOC + 17
      U1 =   A  *E / L(J)
      K(N1,N1) =  U1 * (CX(J)**2) + BB
      K(N1,N2) =  U1 * CX(J) * CY(J)
      K(N1,N3) =  U1 * CX(J) * CZ(J)
      K(N2,N2) =  U1 * (CY(J)**2) + BB
      K(N2,N3) =  U1 * CZ(J) * CY(J)
      K(N3,N3) =  U1 * (CZ(J)**2) + BB
      DO 10 IY = M1 , N1
      DO 10 JY = IY , N1
      K(JY,IY) = K(IY,JY)
10    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine FINDFB will use the finite difference scheme to   *
C   calculate the DSA for the beam element.                           *
C **.*****************************************************************
      SUBROUTINE FINDFB
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,IOPT,INOMP,DISPA,SIGMAA,
     & A1 ,  A2   , A3  ,  AI1 ,  AI2 , AI3 ,
     & TOTLD,TOTLD2,DIN, AR,NOEP  )
      REAL A1(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL TOTLD(1),TOTLD2(1)
      REAL AXF(1),AXF2(1),AR2(1),AR(1),DIN(1)
      IWFF = 4
      PI = ACOS(-1.0)
      CP = .01
      WRITE(IW,*)' NOEP = ',NOEP
      IF(IOPT .EQ. 1 )THEN
      I = INOMP
      AR2(I) = AR(I)
      AR(I) = AR(I) - CP  * AR(I)
10    CONTINUE
      JJ = INOMP
      WRITE(IW,*)' AR(JJ),DIN(JJ) ',AR(JJ),DIN(JJ)
      A1(JJ) = PI * ( (AR(JJ)**2) -  (DIN(JJ)**2) ) / 4.0
      AI1(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 32.0
      AI2(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 64.0
      AI3(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 64.0
      WRITE(IW,*)A1(JJ),AI1(JJ)
      WRITE(IW,*)AI2(JJ),AI3(JJ)
21    CONTINUE
      ELSEIF(IOPT .EQ. 2) THEN
      I = INOMP
      JJ = INOMP
      AR(I) = AR2(I) + CP  * AR2(I)
100   CONTINUE
      A1(JJ) = PI * ( (AR(JJ)**2) -  (DIN(JJ)**2) ) / 4.0
      AI1(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 32.0
      AI2(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 64.0
      AI3(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 64.0
      WRITE(IW,*)A1(JJ),AI1(JJ)
      WRITE(IW,*)AI2(JJ),AI3(JJ)
22    CONTINUE
      DO 20 I = 1 , NTD
      TOTLD2(I) = TOTLD(I)
      WRITE(IW,*)TOTLD(I)
20    CONTINUE
      ELSEIF( IOPT .EQ. 3 ) THEN
      WRITE(IWFF,*)'SENSITIVITY ANALYSIS USING FINITE DIFFRENCE SCHEME'
      WRITE(IWFF ,*)
      WRITE(IWFF,*)' % OF PERTUPATION = ',CP * 100
      J = INOMP
      WRITE(IWFF ,15)INOMP,AR2(J)-CP *AR2(J),AR2(J),AR2(J)+CP *AR2(J)
      AR(J) = AR2(J)
      JJ = J
      A1(JJ) = PI * ( (AR(JJ)**2) -  (DIN(JJ)**2) ) / 4.0
      AI1(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 32.0
      AI2(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 64.0
      AI3(JJ) = PI * ( (AR(JJ)**4) - (DIN(JJ)**4) ) / 64.0
15    FORMAT(/26X,'DESIGN VARIABLE NO:',I5/25X,'----------------'
     &  //2x,' AREA1 =',F10.4,7X,'AREA =',F10.4,10X,' AREA2 =',F10.4)
      WRITE(IWFF ,*)
      WRITE(IWFF ,*)' DEGREE OF FREEDOM          SENSITIVITY'
      WRITE(IWFF ,*)'====================        ==========='
      DO 40 I = 1 , NTD
      TOTLD(I) = TOTLD(I)
      TOTLD2(I) = TOTLD2(I)
      DD2 = .50 * (TOTLD(I) - TOTLD2(I))/(DISPA  * CP * AR2(J) )
      IF(NTD .LT. 10 ) THEN
      WRITE(IWFF ,25)I,DD2
      ELSE
```

```
      IF( I .EQ. 1 ) THEN
      WRITE(IWFF ,25)I,DD2
      ELSEIF(I .EQ. NTD/2 ) THEN
      WRITE(IWFF ,25)I,DD2
      ELSEIF(I .EQ. NTD) THEN
      WRITE(IWFF ,25)I,DD2
      ENDIF
     ENDIF
25    FORMAT(I10,10X,F15.6)
40    CONTINUE
     WRITE(IWFF ,*)
     WRITE(IWFF ,*)' MEMBER NO:                SENSITIVITY'
     WRITE(IWFF ,*)'==================        ==========='
60    CONTINUE
     ENDIF
     RETURN
     END
C **.*****************************************************************
C  This subroutine FINDFF will use the finite difference scheme to   *
C  calculate the DSA for the truss element.                          *
C **.*****************************************************************
     SUBROUTINE FINDFF
    &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
    & NTRMS,NOMP,NNPE,ICHK,IOPT,INOMP,DISPA,SIGMAA,
    & AR,AR2,TOTLD,TOTLD2,AXF,AXF2    )
     REAL  AR(1),AXF(1),AXF2(1),AR2(1),TOTLD(1),TOTLD2(1)
     CP = .010
     IF(IOPT .EQ. 1 )THEN
     I = INOMP
     AR2(I) = AR(I)
     AR(I) = AR(I) - CP  * AR(I)
10    CONTINUE
     ELSEIF(IOPT .EQ. 2) THEN
     I = INOMP
     JJ = INOMP
     AR(I) = AR2(I) + CP  * AR2(I)
100   CONTINUE
     DO 20 I = 1 , NTD
     TOTLD2(I) = TOTLD(I)
20    CONTINUE
     DO 30 I = 1 , NELMNT
     AXF2(I) = AXF(I)
30    CONTINUE
     ELSEIF( IOPT .EQ. 3 ) THEN
     WRITE(7 ,*)' SENSITIVITY ANALYSIS USING FINITE DIFFRENCE SCHEME '
     WRITE(7 ,*)
     WRITE(7,*)' % OF PERTUPATION = ',CP * 100
     J = INOMP
     WRITE(7 ,15)INOMP,AR2(J)-CP *AR2(J),AR2(J),AR2(J)+CP *AR2(J)
     AR(J) = AR2(J)
15    FORMAT(/26X,'DESIGN VARIABLE NO:',I5/25X,'----------'
    &  //2x,' AREA1 =',F10.4,7X,'AREA =',F10.4,10X,' AREA2 =',F10.4)
     WRITE(7 ,*)
     WRITE(7 ,*)' DEGREE OF FREEDOM          SENSITIVITY'
     WRITE(7 ,*)'==================        ==========='
     DO 40 I = 1 , NTD
     TOTLD(I) = TOTLD(I)**2
     TOTLD2(I) = TOTLD2(I)**2
     DD2 = .50 * (TOTLD(I) - TOTLD2(I))/(DISPA  * CP * AR2(J) )
     WRITE(7 ,25)I,DD2
25    FORMAT(I10,10X,F15.6)
40    CONTINUE
     WRITE(7 ,*)
     WRITE(7 ,*)' MEMBER NO:                SENSITIVITY'
     WRITE(7 ,*)'==================        ==========='
     DO 50 I = 1 , NELMNT
     FORCE = .50 * (AXF(I)-AXF2(I)) / (SIGMAA* CP*AR2(J))
     WRITE(7 ,25)I,FORCE
50    CONTINUE
60    CONTINUE
     ENDIF
     RETURN
     END
C **.*****************************************************************
C  This subroutine BINCRM uses the peicewise linear approximation    *
C  method.                                                           *
C **.*****************************************************************
     Forcesub BINCRM (A,KA,W) of NP ident ME
     REAL A(1)
     Shared REAL AA(880000)
     INTEGER KA(1)
     COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
     COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
     COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
     COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
     COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
     COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
     COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
```

```
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
      COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      Shared REAL T1(16),T2(16),T3(16),T4(16),T5(16),T6(16),T7(16)
      Shared REAL TOL,TIMELT(16),TIMEAM(16),TIMEBS(16)
      Shared REAL TTNOR,TDISP(16),TNEW(16),TIMEP
      Shared INTEGER JSTOP,MULT4,MESTOP,LSTOP,NOTIT(20),NUM4
      Shared LOGICAL  TYPE1
      Shared LOGICAL  TYPE2
      Private INTEGER PNUM4,NADYA
      Externf TRUSS,ROWC,DISPF,UBFITS,PBEAM,UBFIB2,PVCONJ
      Externf PVSOR
      End declarations
      CALL CPUTIM(T6(ME))
      TIMELT(ME)  = 0.0
      TIMEBS(ME)  = 0.0
      TIMEAM(ME)  = 0.0
      TOL = EPS1
      MESTOP = 0
      DO 20 PNUM4 = 1 , NLD
      NUM4 = PNUM4
      write(iw,*)'**************',NUM4,'**************'
      Barrier
      End barrier
      JSTOP = 0
      LSTOP = 0
      TDISP(ME) = 0.0
      TTNOR = 0.0
      TNEW(ME)  = 0.0
      DO 90 NADYA = 1 , 1
      Barrier
      End Barrier
      Barrier
      CALL CPUTIM(T1(ME))
      MULT4 = NUM4
      DO 10 ITRQ = K20 , KL20 -1
      A(ITRQ) = 0.0
10    CONTINUE
      CALL CPUTIM(T2(ME))
      End Barrier
      Barrier
      End barrier
      CALL CPUTIM(T4(ME))
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall TRUSS
     &        (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
     & A(K10),A(K11),A(K12),A(K13),A(K20),KA(M1),KA(MI),
     & KA(M2),KA(M3),KA(M3A),KA(M4),KA(M4A),KA(M6),A(K16) )
      ELSEIF ( JFLAG .EQ. 2 ) THEN
      Forcecall PBEAM
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2),KA(M2A) , KA(M3),KA(M3A), KA(M4),KA(M4A),KA(M6),A(K16) )
      ENDIF
      CALL CPUTIM(T5(ME))
      Barrier
      End barrier
      NEQP1 = NTD + 1
      IF(NUM4  .LE. ISWTCH ) THEN
      CALL CPUTIM(T1(ME))
      Forcecall ROWC
     &            (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
     & NTRMS,1,JOPS,ICHK,NUM4,MESTOP)
      CALL CPUTIM(T2(ME))
      Barrier
      TIMEP = 0.0
      write(iw,*)'TIME IN ROWC = ', T2(ME) - T1(ME)
      do 77 ii = 1 , np
      TIMEP =max(timep , T2(ME) -T1(ME) )
77    continue
      write(iw,*) 'Number of Operation Count Fact. = ',JOPS
      FLOPR = JOPS /(T2(ME) - T1(ME))
      write(iw,*) 'Flop Rate In Fact.              = ',FLOPR
      IF(MESTOP .EQ. 1 ) THEN
      WRITE(IW,*) '*****************************************'
      WRITE(IW,*) '* S W I T C H  T O  A R C  L E N G T H *'
      WRITE(IW,*) '*****************************************'
      ENDIF
```

```
      End barrier
      IF(MESTOP .EQ. 1 ) STOP
      CALL CPUTIM(T2(ME))
      Forcecall ROWC
     &            (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK,NUM4,MESTOP)
      CALL CPUTIM(T3(ME))
      Barrier
       timeh = 0.0
       do 78 ii = 1 , np
       TIMEh =max(timeh , T3(ME) -T2(ME) )
78      continue
       write(iw,*)' TIME FOR B.S =  ', T3(ME) - T2(ME)
       FLOPR = JOPS /(T3(ME) - T2(ME))
       write(iw,*) 'Number of Operation Count B.S   = ',JOPS
       write(iw,*) 'Flop Rate In B.S                = ',FLOPR
       timep = timep + timeh
       write(iw,*)'TOTAL TIME IN Choleski = ', TIMEP
      IF(NUM4  .EQ. ISWTCH ) THEN
      REWIND(4)
      WRITE(4,*)TIMEP
      CALL AAAA(A(K20),AA,NTRMS)
      ENDIF
      End barrier
      LIMA = LIMA + 2
      Barrier
      End barrier
      ELSE
      CALL CPUTIM(T1(ME))
      IF(ISOR .EQ. 1 ) THEN
      call cputim(ttt1)
      CALL     PVSOR(IR,IW,NTD,NTRMS,A(K20),A(K14),A(K14A),100,EPS2,
     & KA(M7),KA(M6),ICHK,A(K14B),A(K14C),A(K14D),W)
      call cputim(ttt2)
      ENDIF
      IF(ISOR .EQ. 2 ) THEN
      call cputim(ttt1)
      Forcecall PVCONJ (IR,IW,NTD,NTRMS,A(K20),A(K14),A(K14A),100,EPS2,
     & KA(M7),KA(M6),ICHK,A(K14B),A(K14C),A(K14D),AA,KA(M5))
      Barrier
      End barrier
      call cputim(ttt2)
      ENDIF
      Barrier
      REWIND(4)
      READ(4,*)TIMEP
      TIMEC = TTT2 - TTT1
      write(iw,*)' NUM4  = ',NUM4
      IF(TIMEC .GT. TIMEP) ISWTCH = NUM4  + 1
      write(iw,*)' ISWTCH= ',ISWTCH
      write(iw,*)     ' TIME IN CONJ OR S.O.R = ',TTT2 -TTT1
      write(iw,*)     ' TIME IN CHOL = ',TIMEP
      write(iw,*)' nadya = ',nadya
      End barrier
      CALL CPUTIM(T2(ME))
      CALL CPUTIM(T3(ME))
      ENDIF
      TIMELT(ME) = T2(ME) - T1(ME) + TIMELT(ME)
      TIMEBS(ME) = T3(ME) - T2(ME) + TIMEBS(ME)
      TIMEAM(ME) = T5(ME) - T4(ME) + TIMEAM(ME)
      Barrier
      End barrier
      CALL CPUTIM(T2(ME))
      Forcecall DISPF (IR,IW,NNODE ,A(K14),A(K14A),KA(M1),A(K15),NTD,
     & A(K1),A(K2),A(K3),ICHK,NUM4,ISWTCH)
      CALL CPUTIM(T1(ME))
      TDISP(ME) = TDISP(ME) + T1(ME) - T2(ME)
      Barrier
      End barrier
      CALL CPUTIM(T1(ME))
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall UBFITS (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(K6),A(K7),
     & A(K8),KA(M1),A(K10),A(K11),A(K12),A(K13),A(K1),A(K2),A(K3),
     & KA(M2) ,ICHK ,A(K10A),NUM4,MULT4 ,KA(M2) ,A(K16) ,KA(M1),
     & NNODE,NDPN,NTD,A(K17),A(K14),NADYA,DELMDA,EPS1,METHOD,JSTOP)
      ELSEIF ( JFLAG .EQ. 2 ) THEN
      Forcecall TUBFIB
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MULT4,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K15),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10A) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),A(K14),
     & A(K17),TOL,JSTOP,METHOD,NUM4 ,A(K10) )
      ENDIF
      CALL CPUTIM(T2(ME))
      TNEW(ME) = TNEW(ME) + T2(ME) - T1(ME)
      Barrier
```

```
      NPFLAG = 1
      CALL      WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
     & ICHK,NUM4)
      End Barrier
      IF(JSTOP .EQ. 1 ) GO TO 21
90    CONTINUE
21    CONTINUE
      NOTIT(NUM4) = NADYA
20    CONTINUE
      Barrier
      TMAXLT = 0.0
      TMAXBS = 0.0
      TMAXAM = 0.0
      TMAXUF = 0.0
      TMAXDP = 0.0
      WRITE(IW,*)
      WRITE(IW,*) ' N U M B E R   O F   P R O C E S E S = ', NP
      WRITE(IW,*)
      CALL      WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
     & ICHK,NUM4)
      DO 91 I = 1 , NLD
      NN = NOTIT(I)
      WRITE(IW,*)' Load Level = ',I,'  Number of Iteration =',NN
91    CONTINUE
      DO 100 I = 1 , NP
      TMAXLT = MAX(TMAXLT , TIMELT(I) )
      TMAXBS = MAX(TMAXBS , TIMEBS (I))
      TMAXAM = MAX(TMAXAM , TIMEAM(I) )
      TMAXUF = MAX(TMAXUF , TNEW  (I) )
      TMAXDP = MAX(TMAXDP , TDISP (I) )
      WRITE(IW,*)' Process No. = ',I,' ASSEMBLE    TIME = ',TIMEAM(I)
      WRITE(IW,*)' Process No. = ',I,' FACTORIZE   TIME = ',TIMELT(I)
      WRITE(IW,*)' Process No. = ',I,' B.S.        TIME = ',TIMEBS(I)
      WRITE(IW,*)' Process No. = ',I,' U.B. FORCES TIME = ',TNEW  (I)
      WRITE(IW,*)' Process No. = ',I,' DISP        TIME = ',TDISP (I)
      WRITE(IW,*)
100   CONTINUE
      WRITE(IW,*) ' ASSMEBLE   MAX TIME    = ',TMAXAM
      WRITE(IW,*) ' FACTORIZE  MAX TIME    = ',TMAXLT
      WRITE(IW,*) ' B.S.       MAX TIME    = ',TMAXBS
      WRITE(IW,*) ' U.B. FORCES MAX TIME   = ',TMAXUF
      WRITE(IW,*) ' DISP       MAX TIME    = ',TMAXDP
      IF( NP .EQ. 1) THEN
       WRITE(8) TMAXLT,TMAXBS,TMAXAM
      ELSE
       REWIND(8)
      READ(8) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR ASMBLE     = ',RMAXAM /TMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR L.T.       = ',RMAXLT / TMAXLT
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR B.S        = ',RMAXBS / TMAXBS
      WRITE(IW,*)
      ENDIF
      End Barrier
      CALL CPUTIM(T7(ME))
      TNEWTN = T7(ME) -T6(ME)
      WRITE(IW,*) ' ME = ', ME,'  TOTAL TIME IN NEWTON = ',TNEWTN
      Barrier
      End barrier
95    FORMAT(15X,'********** I T E R A T I O N   N O : ',I5)
195   FORMAT(15X,'..... N E W T O N        N O : ',I5)
999   RETURN
      END
C **.*****************************************************************
C   This subroutine LINEAR is the main subroutine for linear static   *
C   analysis.                                                          *
C **.*****************************************************************
      Forcesub LINEAR (A,KA,W) of NP ident ME
      REAL A(1)
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
      COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34
```

```
      COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
           Shared REAL T1(16),T2(16),T3(16),T4(16),T5(16),T6(16),T7(16)
           Shared REAL TOL,TIMELT(16),TIMEAM(16),TIMEBS(16)
           Shared REAL TTNOR,TDISP(16),TNEW(16),NEQP1
           Shared INTEGER JSTOP,MULT4,MESTOP,LSTOP,NOTIT(20)
           Shared LOGICAL  TYPE1
           Shared LOGICAL  TYPE2
           Private INTEGER NUM4,NADYA
           Externf TRUSS,ROWC,DISPF,UBFITS,PBEAM,UBFIB2
           End declarations
           NEQP1 = NTD + 1
           CALL CPUTIM(T6(ME))
           TIMELT(ME)  = 0.0
           TIMEBS(ME)  = 0.0
           TIMEAM(ME)  = 0.0
           TOL = EPS1
           MESTOP = 0
           DO 20 NUM4 = 1 , NLD
           Barrier
           End barrier
           JSTOP = 0
           LSTOP = 0
           TDISP(ME) = 0.0
           TTNOR = 0.0
           TNEW(ME)  = 0.0
           Barrier
           CALL CPUTIM(T1(ME))
           MULT4 = NUM4 - 1
           End Barrier
           Presched do 10 ITRQ = K20 , KL20 -1
           A(ITRQ) = 0.0
      10   End presched do
           CALL CPUTIM(T2(ME))
           Barrier
           End barrier
           CALL CPUTIM(T4(ME))
           IF( JFLAG .EQ. 1 ) THEN
           Forcecall TRUSS
          &        (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
          & NTRMS,NOMP,NNPE,ICHK,
          & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
          & A(K10),A(K11),A(K12),A(K13),A(K20),KA(M1),KA(MI),
          & KA(M2),KA(M3),KA(M3A),KA(M4),KA(M4A),KA(M6),A(K16),NOOC )
           ELSEIF ( JFLAG .EQ. 2 ) THEN
           Forcecall PBEAM
          &        (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
          & NTRMS,NOMP,NNPE,ICHK,
          & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
          & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
          & A(K10) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
          & KA(M2),KA(M2A) , KA(M3),KA(M3A), KA(M4),KA(M4A),KA(M6),A(K16) )
           ENDIF
           CALL CPUTIM(T5(ME))
           Barrier
           End barrier
           IF(NADYA .LE. ISWTCH ) THEN
           CALL CPUTIM(T1(ME))
           Forcecall ROWC
          &        (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
          & NTRMS,1,JOPS,ICHK,NUM4,MESTOP)
           CALL CPUTIM(T2(ME))
           Barrier
           NOOC = JOPS + NOOC
           End barrier
           Forcecall ROWC
          &        (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
          & NTRMS,2,JOPS,ICHK,NUM4,MESTOP)
           CALL CPUTIM(T3(ME))
           TIMELT(ME) = T2(ME) - T1(ME) + TIMELT(ME)
           TIMEBS(ME) = T3(ME) - T2(ME) + TIMEBS(ME)
           TIMEAM(ME) = T5(ME) - T4(ME) + TIMEAM(ME)
           Barrier
           NOOC = NOOC + JOPS
           End barrier
           CALL CPUTIM(T2(ME))
           Forcecall DISPF (IR,IW,NNODE ,A(K14),A(K14A),KA(M1),A(K15),NTD,
          & A(K1),A(K2),A(K3),ICHK,NUM4,ISWTCH,NOOC)
           CALL CPUTIM(T1(ME))
           TDISP(ME) = TDISP(ME) + T1(ME) - T2(ME)
           Barrier
           End barrier
           CALL CPUTIM(T1(ME))
      C_____________________________________________________________
           IF( JFLAG .EQ. 1 ) THEN
           CALL      AXIAL ( IR,IW,NNODE,NELMNT,NNPE,A(K14),KA(M1),KA(MI),
          & ICHK,NDPN,A(K11),A(K12),A(K13),A(K4),A(K7),KA(M2),A(K10),A(K16),
          & A(K17),NTD )
```

```
      ELSEIF( JFLAG .EQ. 2 ) THEN
      CALL
     &       BEAMST (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K14),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6))
C_____________________________________________________
      CALL CPUTIM(T2(ME))
      TNEW(ME) = TNEW(ME) + T2(ME) - T1(ME)
      Barrier
      NPFLAG = 1
      CALL      WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
     & ICHK)
      End Barrier
20    CONTINUE
      CALL CPUTIM(T7(ME))
      Barrier
      TMAXLT = 0.0
      TMAXBS = 0.0
      TMAXAM = 0.0
      TMAXUF = 0.0
      TMAXDP = 0.0
      WRITE(IW,*)
      WRITE(IW,*) ' N U M B E R   O F   P R O C E S E S = ', NP
      WRITE(IW,*)
      CALL      WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
     & ICHK)
      DO 91 I = 1 , NLD
      NN = NOTIT(I)
      WRITE(IW,*)' Load Level = ',I,'   Number of Iteration =',NN
91    CONTINUE
      DO 100 I = 1 , NP
      TMAXLT = MAX(TMAXLT , TIMELT(I) )
      TMAXBS = MAX(TMAXBS , TIMEBS (I))
      TMAXAM = MAX(TMAXAM , TIMEAM(I) )
      TMAXUF = MAX(TMAXUF , TNEW  (I) )
      TMAXDP = MAX(TMAXDP , TDISP (I) )
      WRITE(IW,*)' Process No. = ',I,' ASSEMBLE    TIME = ',TIMEAM(I)
      WRITE(IW,*)' Process No. = ',I,' FACTORIZE   TIME = ',TIMELT(I)
      WRITE(IW,*)' Process No. = ',I,' B.S.        TIME = ',TIMEBS(I)
      WRITE(IW,*)' Process No. = ',I,' U.B. FORCES TIME = ',TNEW  (I)
      WRITE(IW,*)' Process No. = ',I,' DISP        TIME = ',TDISP (I)
      WRITE(IW,*)
100   CONTINUE
      WRITE(IW,*) ' ASSMEBLE    MAX TIME    = ',TMAXAM
      WRITE(IW,*) ' FACTORIZE   MAX TIME    = ',TMAXLT
      WRITE(IW,*) ' B.S.        MAX TIME    = ',TMAXBS
      WRITE(IW,*) ' U.B. FORCES MAX TIME    = ',TMAXUF
      WRITE(IW,*) ' DISP        MAX TIME    = ',TMAXDP
      IF( NP .EQ. 1) THEN
       WRITE(8) TMAXLT,TMAXBS,TMAXAM
      ELSE
       REWIND(8)
      READ(8) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR ASMBLE     = ',RMAXAM /TMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR L.T.       = ',RMAXLT / TMAXLT
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR B.S        = ',RMAXBS / TMAXBS
      WRITE(IW,*)
      ENDIF
      End Barrier
      CALL CPUTIM(T7(ME))
      TNEWTN = T7(ME) -T6(ME)
      WRITE(IW,*) ' ME = ', ME,'   TOTAL TIME IN NEWTON = ',TNEWTN
      Barrier
      End barrier
95    FORMAT(15X,'********** I T E R A T I O N   N O : ',I5)
195   FORMAT(15X,'..... N E W T O N        N O : ',I5)
999   RETURN
      END
C **.*****************************************************************
C  This subroutine MATM used to multiply the T marix by the Kii      *
C  for the beam element.                                             *
C **.*****************************************************************
      SUBROUTINE MATM(N1,N2,N3,A,B,C)
      REAL  A(N1,N2),B(N2,N3),C(N1,N3)
      DO 30 K = 1 , 6
      C(K,1) =   A(K,1) * B(1,1)
     &         + A(K,2) * B(2,1)
     &         + A(K,3) * B(3,1)
      C(K,2) =   A(K,1) * B(1,2)
     &         + A(K,2) * B(2,2)
```

```
     &         + A(K,3) * B(3,2)
      C(K,3) =   A(K,1) * B(1,3)
     &         + A(K,2) * B(2,3)
     &         + A(K,3) * B(3,3)
30    CONTINUE
      DO 130 K = 1 , 6
      C(K,4) =    A(K,4) * B(4,4)
     &          + A(K,5) * B(5,4)
     &          + A(K,6) * B(6,4)
      C(K,5) =    A(K,4) * B(4,5)
     &          + A(K,5) * B(5,5)
     &          + A(K,6) * B(6,5)
      C(K,6) =    A(K,4) * B(4,6)
     &          + A(K,5) * B(5,6)
     &          + A(K,6) * B(6,6)
130   CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine MATM2  used to multiply the matrix Kjj by the     *
C  T matrix for the beam element.                                    *
C **.*****************************************************************
      SUBROUTINE MATM2(N1,N2,N3,A,B,C)
      REAL  A(N1,N2),B(N2,N3),C(N1,N3)
      DO 30 K = 1 , 6
      C(1,K) =    A(1,1) * B(1,K)
     &          + A(1,2) * B(2,K)
     &          + A(1,3) * B(3,K)
      C(2,K) =    A(2,1) * B(1,K)
     &          + A(2,2) * B(2,K)
     &          + A(2,3) * B(3,K)
      C(3,K) =    A(3,1) * B(1,K)
     &          + A(3,2) * B(2,K)
     &          + A(3,3) * B(3,K)
30    CONTINUE
      DO 130 K = 1 , 6
      C(4,K) =    A(4,4) * B(4,K)
     &          + A(4,5) * B(5,K)
     &          + A(4,6) * B(6,K)
      C(5,K) =    A(5,4) * B(4,K)
     &          + A(5,5) * B(5,K)
     &          + A(5,6) * B(6,K)
      C(6,K) =    A(6,4) * B(4,K)
     &          + A(6,5) * B(5,K)
     &          + A(6,6) * B(6,K)
130   CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine MATM3 used to multiply the element matrix K by    *
C  the T matrix.                                                     *
C **.*****************************************************************
      SUBROUTINE MATM3 (N1,N2,N3,A,B,C)
      REAL  A(N1,N2),B(N2,N3),C(N1,N3)
      DO 30 K = 1 , 12
      DO 20 I = 1 , 12, 3
      C(K,I  ) =    A(K,I  ) * B(1,1)
     &          +  A(K,I+1) * B(2,1)
     &          +  A(K,I+2) * B(3,1)
      C(K,I+1) =    A(K,I  ) * B(1,2)
     &          +  A(K,I+1) * B(2,2)
     &          +  A(K,I+2) * B(3,2)
      C(K,I+2) =    A(K,I  ) * B(1,3)
     &          +  A(K,I+1) * B(2,3)
     &          +  A(K,I+2) * B(3,3)
20    CONTINUE
30    CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine MATM4 used to multiply the matrix K by the vector *
C  T for the beam element.                                           *
C **.*****************************************************************
      SUBROUTINE MATM4 (N1,N2,N3,A,B,C)
      REAL  A(N1,N2),B(N2,N3),C(N1,N3)
      DO 30 K = 1 , 12
      DO 20 I = 1 , 12, 3
      C(I,K  ) =    A(1,1) * B(I  ,K)
     &          +  A(1,2) * B(I+1,K)
     &          +  A(1,3) * B(I+2,K)
      C(I+1,K) =    A(2,1) * B(I  ,K)
     &          +  A(2,2) * B(I+1,K)
     &          +  A(2,3) * B(I+2,K)
      C(I+2,K) =    A(3,1) * B(I  ,K)
     &          +  A(3,2) * B(I+1,K)
     &          +  A(3,3) * B(I+2,K)
20    CONTINUE
30    CONTINUE
```

```
      RETURN
      END
C **.*****************************************************************
C  This subroutine MATMUL used to multiply matrix by vector.          *
C **.*****************************************************************
      SUBROUTINE MATMUL(N1,N2,N3,A,B,C)
      REAL  A(N1,N2),B(N2,N3),C(N1,N3)
      DO 30 K = 1 , N1
      DO 20 I = 1 , N3
      SUM = 0.0
      DO 10 J = 1 , N2
      SUM = SUM + A(K,J) * B(J,I)
10    CONTINUE
      C(K,I) = SUM
20    CONTINUE
30    CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine MAXJNE used to find the maximum number of elements*
C  connected to a node.                                               *
C **.*****************************************************************
      SUBROUTINE MAXJNE (IR,IW,NE,NJ,NODELM,MAXJTM,NUIT,
     & NNPE,LINDA,ICHK )
      INTEGER NODELM(NNPE,NE),NUIT(NJ)
      MAXJTM = 0
      DO 10 I = 1 , NJ
      NUIT(I) = 0
10    CONTINUE
      DO 20 I = 1 , NE
      II = NODELM(LINDA,I )
      NUIT(II) = NUIT(II) + 1
      MAXJTM = MAX ( MAXJTM , NUIT(II)  )
20    CONTINUE
      WRITE(IW,15) MAXJTM
15    FORMAT(2X,' MAX JOINT MEMBERS =',I4 )
      RETURN
      END
C **.*****************************************************************
C  This subroutine MNEWTN used for nonlinear F.E.E using mN-R         *
C  Method.                                                            *
C **.*****************************************************************
      Forcesub MNEWTN (A,KA,W,NOOCT) of NP ident ME
      REAL A(1)
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
      COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      Shared REAL T1(16),T2(16),T3(16),T4(16),T5(16),T6(16),T7(16)
      Shared REAL TOL,TIMELT(16),TIMEAM(16),TIMEBS(16)
      Shared REAL TTNOR,TDISP(16),TNEW(16)
      Shared INTEGER NOOCA,NOOCB,NOOCC,NOOCU,NOOCD
      Shared INTEGER JSTOP,MULT4, MESTOP,LSTOP,NOTIT(50)
      Shared LOGICAL TYPE1
      Private INTEGER NUM4,NADYA
      Externf TRUSS1,PBEAM,UBFITS,DISPF,ROWC
      End declarations
      CALL CPUTIM(T6(ME))
      TIMELT(ME)  = 0.0
      TIMEBS(ME)  = 0.0
      TIMEAM(ME)  = 0.0
      TOL = EPS1
      NOOCA = 0
      NOOCB = 0
      NOOCC = 0
      NOOCU = 0
      NOOCD = 0
      DO 20 NUM4 = 1 , NLD
      Barrier
      End barrier
      MAXNIT = 100
      Critical TYPE1
      JSTOP = 0
      LSTOP = 0
      TDISP(ME) = 0.0
      TTNOR = 0.0
```

```
      TNEW(ME) = 0.0
      End critical
      DO 90 NADYA = 1 , MAXNIT
      Barrier
      End barrier
      Barrier
      MULT4 = NUM4 - 1
      End Barrier
      Barrier
      End barrier
      IF(NADYA .GT. 2 ) GO TO 301
      Barrier
      DO 101 ITRQ = K20 , KL20 -1
      A(ITRQ) = 0.0
101   CONTINUE
      End barrier
      Barrier
      End barrier
      CALL CPUTIM(T4(ME))
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall TRUSS
     &        (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
     & A(K10),A(K11),A(K12),A(K13),A(K20),KA(M1),KA(MI),
     & KA(M2),KA(M3),KA(M3A),KA(M4),KA(M4A),KA(M6),A(K16),NOOCA )
      ELSEIF ( JFLAG .EQ. 2 ) THEN
      Forcecall PBEAM
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2),KA(M2A),KA(M3),KA(M3A), KA(M4),KA(M4A),KA(M6),A(K16),
     & NOOCA)
      ENDIF
      CALL CPUTIM(T5(ME))
      NEQP1 = NTD + 1
      JOPS = 0
      Barrier
      End barrier
      CALL CPUTIM(T1(ME))
      Forcecall ROWC
     &       (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
     & NTRMS,1,JOPS,ICHK,NUM4,MESTOP)
      CALL CPUTIM(T2(ME))
      Barrier
      End barrier
      Critical TYPE1
      write(iw,*) ' subroutine rowc is used'
      TIMELT(ME) = T2(ME) - T1(ME) + TIMELT(ME)
      TIMEAM(ME) = T5(ME) - T4(ME) + TIMEAM(ME)
      NOOCC = NOOCC + JOPS
      End critical
      Barrier
      IF(MESTOP .EQ. 1 ) THEN
      WRITE(IW,*) '*****************************************'
      WRITE(IW,*) '* S W I T C H  T O  A R C  L E N G T H *'
      WRITE(IW,*) '*****************************************'
      ENDIF
      End barrier
      IF(MESTOP .EQ. 1 ) STOP
301   NEQP1 = NTD + 1
      CALL CPUTIM(T2(ME))
      Barrier
      End barrier
      JOPS = 0
      Forcecall ROWC
     &       (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK,NUM4,MESTOP)
      Barrier
      End barrier
      CALL CPUTIM(T3(ME))
      Critical TYPE1
      TIMEBS(ME) = T3(ME) - T2(ME) + TIMEBS(ME)
      NOOCB = NOOCB + JOPS
      End critical
      Barrier
      End barrier
      CALL CPUTIM(T2(ME))
      Forcecall DISPF (IR,IW,NNODE ,A(K14),A(K14A),KA(M1),A(K15),NTD,
     & A(K1),A(K2),A(K3),ICHK,NUM4,ISWTCH,NOOCD)
      CALL CPUTIM(T1(ME))
      TDISP(ME) = TDISP(ME) + T1(ME) - T2(ME)
      Barrier
      End barrier
      CALL CPUTIM(T1(ME))
      IF( JFLAG .EQ. 1 )THEN
```

```
      Forcecall UBFITS (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(K6),A(K7),
     & A(K8),KA(MI),A(K10),A(K11),A(K12),A(K13),A(K1),A(K2),A(K3),
     & KA(M2) ,ICHK ,A(K10A),NUM4,MULT4 ,KA(M2) ,A(K16) ,KA(M1),
     & NNODE,NDPN,NTD,A(K17),A(K14),NADYA,DELMDA,EPS1,METHOD,JSTOP,
     & NOOCU)
      ELSEIF( JFLAG .EQ. 2 )THEN
      Forcecall UBFIB2
     &         (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MULT4,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K15),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10A) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),A(K14),
     & A(K17),TOL,JSTOP,METHOD,NUM4 ,A(K10),NOOCU)
      ENDIF
      CALL CPUTIM(T2(ME))
      TNEW(ME) = TNEW(ME) + T2(ME) - T1(ME)
      Barrier
      End barrier
      IF(JSTOP .EQ. 1 ) GO TO 21
90    CONTINUE
21    CONTINUE
      NOTIT(NUM4) = NADYA
20    CONTINUE
      CALL CPUTIM(T7(ME))
      Barrier
      TMAXLT = 0.0
      TMAXBS = 0.0
      TMAXAM = 0.0
      TMAXUF = 0.0
      TMAXDP = 0.0
      WRITE(IW,*)
      WRITE(IW,*) ' N U M B E R   O F   P R O C E S E S = ', NP
      WRITE(IW,*)
      CALL      WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
     & ICHK)
      WRITE(IW,*)'*********************************************'
      WRITE(IW,*)' Assembly Operation Count      = ',NOOCA
      WRITE(IW,*)' Factorization Operation Count = ',NOOCC
      WRITE(IW,*)' B. Substitution Operation Count = ',NOOCB
      WRITE(IW,*)' U.B.Forces      Operation Count = ',NOOCU
      WRITE(IW,*)' Displacement    Operation Count = ',NOOCD
      WRITE(IW,*)'*********************************************'
      DO 91 I = 1 , NLD
      NN = NOTIT(I)
      WRITE(IW,*)' Load Level = ',I,'   Number of Iteration =',NN
91    CONTINUE
      DO 100 I = 1 , NP
      TMAXLT = MAX(TMAXLT , TIMELT(I) )
      TMAXBS = MAX(TMAXBS , TIMEBS (I))
      TMAXAM = MAX(TMAXAM , TIMEAM(I) )
      TMAXUF = MAX(TMAXUF , TNEW  (I) )
      TMAXDP = MAX(TMAXDP , TDISP (I) )
      WRITE(IW,*)' Process No. = ',I,' ASSEMBLE    TIME = ',TIMEAM(I)
      WRITE(IW,*)' Process No. = ',I,' L.T.        TIME = ',TIMELT(I)
      WRITE(IW,*)' Process No. = ',I,' B.S.        TIME = ',TIMEBS(I)
      WRITE(IW,*)' Process No. = ',I,' U.B. FORCES TIME = ',TNEW  (I)
      WRITE(IW,*)' Process No. = ',I,' DISP        TIME = ',TDISP (I)
      WRITE(IW,*)
100   CONTINUE
      WRITE(IW,*) '============================='
      WRITE(IW,*) '=======     Max Time Calculations    ============'
      WRITE(IW,*) '============================='
      WRITE(IW,*)
      WRITE(IW,*) ' ASSMEBLE    MAX TIME    = ',TMAXAM
      WRITE(IW,*) ' FACTORIZE   MAX TIME    = ',TMAXLT
      WRITE(IW,*) ' B.S.        MAX TIME    = ',TMAXBS
      WRITE(IW,*) ' U.B. FORCES MAX TIME    = ',TMAXUF
      WRITE(IW,*) ' DISP        MAX TIME    = ',TMAXDP
      WRITE(IW,*) '====================================='
      WRITE(IW,*) '=======  Mega Flop Rate Calculations  =========='
      WRITE(IW,*) '====================================='
      WRITE(IW,*)
      WRITE(IW,*) ' ASSMEBLE    FLOP RATE   = ',NOOCA /(TMAXAM*1.0E06)
      WRITE(IW,*) ' FACTORIZE   FLOP RATE   = ',NOOCC /(TMAXLT*1.0E06)
      WRITE(IW,*) ' B.S.        FLOP RATE   = ',NOOCB /(TMAXBS*1.0E06)
      WRITE(IW,*) ' U.B. FORCES FLOP RATE   = ',NOOCU/(TMAXUF *1.0E06)
      WRITE(IW,*) ' DISP        FLOP RATE   = ',NOOCD/(TMAXDP *1.0e06)
      WRITE(IW,*) '============================='
      NOOCT = NOOCA + NOOCB + NOOCC + NOOCU + NOOCD
      IF( NP .EQ. 1) THEN
       WRITE(2) TMAXLT,TMAXBS,TMAXAM
      ELSE
       REWIND(2)
      READ(2) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
```

```
      WRITE(IW,*) 'SPEED UP FOR ASSEMBLE   = ',RMAXAM /TMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR L.T.       = ',RMAXLT / TMAXLT
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR B.S        = ',RMAXBS / TMAXBS
      WRITE(IW,*)
      ENDIF
      End Barrier
      CALL CPUTIM(T7(ME))
      Critical TYPE1
      TNEWTN = T7(ME) -T6(ME)
      WRITE(IW,*) ' ME = ', ME,'   TOTAL TIME IN M. NEWTON = ',TNEWTN
      End critical
      Barrier
      End barrier
95    FORMAT(15X,'********** I T E R A T I O N   N O : ',I5)
195   FORMAT(15X,'..... N E W T O N        N O : ',I5)
999   RETURN
      END
C **.*****************************************************************
C  This subroutine NEWID used to find the elements connected to a    *
C  node.                                                             *
C **.*****************************************************************
      SUBROUTINE NEWID  (IR,IW,NE,NJ,NODELM,MAXJTN,MT,NUIT,
     & NNPE,LINDA,ICHK)
      INTEGER  NODELM(NNPE,NE),MT(MAXJTN,NJ),NUIT(NJ)
      DO  1  I = 1, NJ
      NUIT(I) = 0
      DO  1  J = 1, MAXJTN
1     MT(J,I) = 0
      DO 10 I = 1 , NE
      II = NODELM(LINDA,I)
      MT(  NUIT(II)+1,II) = I
      NUIT(II) = NUIT(II) + 1
10    CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine NEWTON used for nonlinear F.E.A. using N-R method *
C **.*****************************************************************
      Forcesub NEWTON (A,KA,W) of NP ident ME
      REAL A(1)
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
      COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      Shared REAL T1(16),T2(16),T3(16),T4(16),T5(16),T6(16),T7(16)
      Shared REAL TOL,TIMELT(16),TIMEAM(16),TIMEBS(16)
      Shared REAL TTNOR,TDISP(16),TNEW(16),NEQP1
      Shared INTEGER JSTOP,MULT4,MESTOP,LSTOP,NOTIT(20)
      Shared LOGICAL  TYPE1
      Shared LOGICAL  TYPE2
      Private INTEGER NUM4,NADYA
      Externf TRUSS,ROWC,DISPF,UBFITS,PBEAM,UBFIB2
      End declarations
      NEQP1 = NTD + 1
      CALL CPUTIM(T6(ME))
      TIMELT(ME)  = 0.0
      TIMEBS(ME)  = 0.0
      TIMEAM(ME)  = 0.0
      TOL = EPS1
      MESTOP = 0
      DO 20 NUM4 = 1 , NLD
      Barrier
      End barrier
      JSTOP = 0
      LSTOP = 0
      TDISP(ME) = 0.0
      TTNOR = 0.0
      TNEW(ME)  = 0.0
      DO 90 NADYA = 1 , MAXNIT
      Barrier
      End Barrier
      Barrier
      CALL CPUTIM(T1(ME))
      MULT4 = NUM4 - 1
```

```
      End Barrier
      Presched do 10 ITRQ = K20 , KL20 -1
      A(ITRQ) = 0.0
10    End presched do
      CALL CPUTIM(T2(ME))
      Barrier
      End barrier
      CALL CPUTIM(T4(ME))
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall TRUSS
     &        (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
     & A(K10),A(K11),A(K12),A(K13),A(K20),KA(M1),KA(MI),
     & KA(M2),KA(M3),KA(M3A),KA(M4),KA(M4A),KA(M6),A(K16),NOOC )
      ELSEIF ( JFLAG .EQ. 2 ) THEN
      Forcecall PBEAM
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2),KA(M2A) , KA(M3),KA(M3A), KA(M4),KA(M4A),KA(M6),A(K16) )
      ENDIF
      CALL CPUTIM(T5(ME))
      Barrier
      End barrier
      IF(NADYA .LE. ISWTCH ) THEN
      CALL CPUTIM(T1(ME))
      Forcecall ROWC
     &           (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
     & NTRMS,1,JOPS,ICHK,NUM4,MESTOP)
      CALL CPUTIM(T2(ME))
      Barrier
      NOOC = JOPS + NOOC
      End barrier
      Barrier
      IF(MESTOP .EQ. 1 ) THEN
      WRITE(IW,*) '*****************************************'
      WRITE(IW,*) '* SWITCH TO ARC LENGTH *'
      WRITE(IW,*) '*****************************************'
      ENDIF
      End barrier
      IF(MESTOP .EQ. 1 ) STOP
      Forcecall ROWC
     &           (A(K20),A(K14),KA(M6),KA(M7),KA(M5),NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK,NUM4,MESTOP)
      CALL CPUTIM(T3(ME))
      ELSE
      IF(ISOR .EQ. 1 ) THEN
      CALL      SOR2(IR,IW,NTD,NTRMS,A(K20),A(K14),A(K14A),100,EPS2,
     & KA(M7),KA(M6),ICHK,A(K14B),A(K14C),A(K14D),W)
      ELSEIF(ISOR .EQ. 2 ) THEN
      CALL      CONJ (IR,IW,NTD,NTRMS,A(K20),A(K14),A(K14A),100,EPS2,
     & KA(M7),KA(M6),ICHK,A(K14B),A(K14C),A(K14D))
      ENDIF
      ENDIF
      TIMELT(ME) = T2(ME) - T1(ME) + TIMELT(ME)
      TIMEBS(ME) = T3(ME) - T2(ME) + TIMEBS(ME)
      TIMEAM(ME) = T5(ME) - T4(ME) + TIMEAM(ME)
      Barrier
      NOOC = NOOC + JOPS
      End barrier
      CALL CPUTIM(T2(ME))
      Forcecall DISPF (IR,IW,NNODE ,A(K14),A(K14A),KA(M1),A(K15),NTD,
     & A(K1),A(K2),A(K3),ICHK,NUM4,ISWTCH,NOOC)
      CALL CPUTIM(T1(ME))
      TDISP(ME) = TDISP(ME) + T1(ME) - T2(ME)
      Barrier
      End barrier
      CALL CPUTIM(T1(ME))
      IF( JFLAG .EQ. 1 ) THEN
      Forcecall UBFITS (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(K6),A(K7),
     & A(K8),KA(MI),A(K10),A(K11),A(K12),A(K13),A(K1),A(K2),A(K3),
     & KA(M2) ,ICHK ,A(K10A),NUM4,MULT4 ,KA(M2) ,A(K16) ,KA(M1),
                                                                    &
NNODE,NDPN,NTD,A(K17),A(K14),NADYA,DELMDA,EPS1,METHOD,JSTOP,NOOC)
      ELSEIF ( JFLAG .EQ. 2 ) THEN
      Forcecall UBFIB2
     &        (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MULT4,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K15),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10A) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),A(K14),
     & A(K17),TOL,JSTOP,METHOD,NUM4 ,A(K10) )
      ENDIF
      CALL CPUTIM(T2(ME))
```

```
      TNEW(ME) = TNEW(ME) + T2(ME) - T1(ME)
      Barrier
      NPFLAG = 1
      End Barrier
      IF(JSTOP .EQ. 1 ) GO TO 21
 90    CONTINUE
 21    CONTINUE
      NOTIT(NUM4) = NADYA
 20    CONTINUE
      CALL CPUTIM(T7(ME))
      Barrier
      TMAXLT = 0.0
      TMAXBS = 0.0
      TMAXAM = 0.0
      TMAXUF = 0.0
      TMAXDP = 0.0
      WRITE(IW,*)
      WRITE(IW,*) ' N U M B E R   O F   P R O C E S E S = ', NP
      WRITE(IW,*)
      CALL      WRITEF (IR,IW,NELMNT,A(K16),IPRINT,NPFLAG,A(K15),NTD,
     & ICHK)
      DO 91 I = 1 , NLD
      NN = NOTIT(I)
      WRITE(IW,*)' Load Level = ',I,'  Number of Iteration =',NN
 91    CONTINUE
      DO 100 I = 1 , NP
      TMAXLT = MAX(TMAXLT , TIMELT(I) )
      TMAXBS = MAX(TMAXBS , TIMEBS (I))
      TMAXAM = MAX(TMAXAM , TIMEAM(I) )
      TMAXUF = MAX(TMAXUF , TNEW  (I) )
      TMAXDP = MAX(TMAXDP , TDISP (I) )
      WRITE(IW,*)' Process No. = ',I,' ASSEMBLE   TIME = ',TIMEAM(I)
      WRITE(IW,*)' Process No. = ',I,' FACTORIZE  TIME = ',TIMELT(I)
      WRITE(IW,*)' Process No. = ',I,' B.S.       TIME = ',TIMEBS(I)
      WRITE(IW,*)' Process No. = ',I,' U.B. FORCES TIME = ',TNEW  (I)
      WRITE(IW,*)' Process No. = ',I,' DISP       TIME = ',TDISP (I)
      WRITE(IW,*)
 100   CONTINUE
      WRITE(IW,*) ' ASSMEBLE   MAX TIME   = ',TMAXAM
      WRITE(IW,*) ' FACTORIZE  MAX TIME   = ',TMAXLT
      WRITE(IW,*) ' B.S.       MAX TIME   = ',TMAXBS
      WRITE(IW,*) ' U.B. FORCES MAX TIME  = ',TMAXUF
      WRITE(IW,*) ' DISP       MAX TIME   = ',TMAXDP
      IF( NP .EQ. 1) THEN
       WRITE(8) TMAXLT,TMAXBS,TMAXAM
      ELSE
       REWIND(8)
      READ(8) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) RMAXLT,RMAXBS,RMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR ASMBLE    = ',RMAXAM /TMAXAM
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR L.T.      = ',RMAXLT / TMAXLT
      WRITE(IW,*)
      WRITE(IW,*) 'SPEED UP FOR B.S       = ',RMAXBS / TMAXBS
      WRITE(IW,*)
      ENDIF
      End Barrier
      CALL CPUTIM(T7(ME))
      TNEWTN = T7(ME) -T6(ME)
      WRITE(IW,*) ' ME = ', ME,'  TOTAL TIME IN NEWTON = ',TNEWTN
      Barrier
      End barrier
 95    FORMAT(15X,'********** I T E R A T I O N  N O : ',I5)
 195   FORMAT(15X,'..... N E W T O N      N O : ',I5)
 999   RETURN
      END
C **.*****************************************************************
C  This subroutine NSAFT  will find the sensitivity of truss elemnt  *
C  with respect to the area.                                         *
C  NSAFT: Nonlinear Sinsitivity Analysis For Truss.                  *
C **.*****************************************************************
      Forcesub NSAFT
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM,  CF,  DM,  AR,WEIGHT,
     & AL  ,CX  ,CY  ,CZ  ,BIGK ,ID   ,NODELM   ,
     & MID  ,NID  ,NJPROW,MAXA,AXF,TOTLD,DKDB,NDV,DG,IROWL,ICOLH,BL,
     & SIGMAA,NDC,NSC,NTDC,DEFA)
     & of NP ident ME
      REAL  X(1),Y(1),Z(1),YM(1),CF(1),DM(1),AR(1),WEIGHT(1),CCX(3)
      REAL  BIGK(1),AXF(1),TOTLD(1),DG(NDV,NTDC)
      REAL  AL(1),CX(1),CY(1),CZ(1),DKDB(NDV,NTD),DCOL(3)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24),ICOLH(1),IROWL(1)
      Private REAL TEMP1(9000),TEMP2(9000),SUM
      Shared REAL T1(16),T2(16),T3(16)
```

```
      Private INTEGER ICONT
      End declarations
C-----------------------------------------------------------------*
      CALL CPUTIM(T1(ME))
      Presched do 3 I = 1 , NDV
      DO 2 J = 1 , NTD
      DKDB(I,J) = 0.0
2     CONTINUE
3     End presched do
      Barrier
      WRITE(9 ,*)
      WRITE(9,*)' SENSITIVITY ANALYSIS USING AJOINT METHOD IN NSAFT '
      ICONT = 0
      End barrier
      Presched do 90 NADYA = 1 , NDV
      DO 30 MAN = 1 , NELMNT
      IF( NADYA .EQ.  MID( MAN ) ) THEN
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK2,MAN,
     & NDPN )
      J = MAN
      PP = AXF(MAN)
      DCOL(1) = CX(J)
      DCOL(2) = CY(J)
      DCOL(3) = CZ(J)
      DO 20 II = 1 , 3
      JJ = LM(II)
      MM = LM(II+3)
      IF(JJ . NE. 0 ) THEN
      DKDB(NADYA,JJ) = DKDB(NADYA,JJ) + PP * DCOL(II)
      ENDIF
      IF( MM .NE. 0 ) THEN
      DKDB(NADYA,MM) = DKDB(NADYA,MM) - PP * DCOL(II)
      ENDIF
20    CONTINUE
      ENDIF
30    CONTINUE
90    End presched do
      Barrier
      End barrier
      ICONT = ICONT + 1
      Presched do 70 I     = 1,NTD
      DO 40 J = 1 , NTD
      IF(I .EQ. J ) THEN
      TEMP1(J) = 2.0* TOTLD(I) / DEFA
      ELSE
      TEMP1(J) = 0.0
      ENDIF
40    CONTINUE
      NEQP1 = NTD + 1
      CALL
     &      SOLVER (BIGK  ,TEMP1 ,MAXA  ,IROWL ,ICOLH ,NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK,1,1)
      DO 60 M = 1 , NDV
      SUM = 0.0
      DO 50 J = 1 , NTD
      SUM = SUM + DKDB(M,J) * TEMP1(J)
50    CONTINUE
      DG(M,I) =  SUM
60    CONTINUE
70    End presched do
      Barrier
      End barrier
C *** THIS WILL FIND THE DERIVITAVES WITH RESPECT TO THE STRESSES
      Presched do 140 MCONT  = 1 , NDV
      ICONT = NDC
      DO 110 II = 1 , NTD
      TEMP2(II) = DG(MCONT,II)  / (2.0 * TOTLD(II) / DEFA )
110   CONTINUE
      DO 130 MAN = 1 , NELMNT
      ICONT = ICONT + 1
      CCX(1) = CX(MAN)
      CCX(2) = CY(MAN)
      CCX(3) = CZ(MAN)
      E =    YM( MID( MAN ) )
      ALL = AL(MAN)
      AREA = AR ( MID(MAN) )
      CALL      SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      SUM = 0.0
      DO 120 NANY = 1 , 3
      II = LM(NANY)
      JJ = LM(NANY+3)
      IF( II .NE. 0 ) THEN
      SUM = SUM  + TEMP2(II) *  CCX(NANY) * E / ( ALL * SIGMAA )
      ENDIF
      IF( JJ .NE. 0 ) THEN
      SUM = SUM   - TEMP2(JJ) *  CCX(NANY) * E /( ALL * SIGMAA )
      ENDIF
```

```
120   CONTINUE
      DG(MCONT,ICONT) = - SUM
130   CONTINUE
140   End presched do
      CALL CPUTIM(T2(ME))
      Barrier
      TMAX = 0.0
      DO 200  I = 1 , NP
      TMAX = MAX(  TMAX,( T2(I) - T1(I)) )
200   CONTINUE
      WRITE(IW,*) ' MAX TIME FOR SENSITIVITY ANALYSIS = ',TMAX
      End barrier
15    FORMAT(6E12.4)
45    FORMAT(5F12.5/5F12.5/5F12.5/5F12.5)
85    FORMAT(I10,10X,F15.6)
95    FORMAT(/26X,'DESIGN VARIABLE NO:',I5/25X,'----------')
     & //)
      RETURN
      END
C **.*****************************************************************
C   This subroutine OCORD used with the D.S.A in the F.D.S in order   *
C   to set the coordinates to the original state.                     *
C **.*****************************************************************
      SUBROUTINE OCORD(IR,IW,NNODE ,D,ID,TOTLD,NTD,X,Y,Z,ICHK)
      REAL D(1),DD(6),TOTLD(1),X(1),Y(1),Z(1),DLD(1)
      INTEGER ID(6,NNODE)
      REWIND(1)
      DO 100 I = 1 , NNODE
      READ(1)XX,YY,ZZ
      X(I) = XX
      Y(I) = YY
      Z(I) = ZZ
100   CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine OLOAD used in the D.S.A with the F.D.S in order   *
C   start the load at the original state.                             *
C **.*****************************************************************
      SUBROUTINE OLOAD  (IR,IW,NTD,NELMNT,D,P,AXF,TOTLD)
      REAL D(1),P(1),AXF(1),TOTLD(1)
      DO 10 I = 1 , NTD
      D(I) = P(I)
      TOTLD(I) = 0.0
10    CONTINUE
      DO 20 I = 1 , NELMNT
      AXF(I) = 0.0
20    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine OLOADB used with D.S.A with the F.D.S in order to *
C   start the load at the original state and set all the original     *
C   data for the beam element.                                        *
C **.*****************************************************************
      SUBROUTINE OLOADB
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM ,PR ,  DM,WEIGHT,
     & A1 ,  A2   , A3  ,  AI1 ,  AI2 , AI3 ,
     & AL   ,C ,BIGK ,ID    ,NODELM ,MID,MEID,NID,NJPROW,MAXA,AXF,
     & D,TOTLD,ALO,DELP )
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL X(1),Y(1),Z(1),YM(1),PR(1),DM(1),WEIGHT(1)
      REAL A1(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL BIGK(1),AXF(1),D(1),TOTLD(1)
      REAL AL(1),C(9,NELMNT),SMLK(12,12),ALO(1),DELP(1)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24),MEID(1)
      DO 10 I = 1 , NTD
      D(I) = DELP(I)
      TOTLD(I) = 0.0
10    CONTINUE
      DO 20 J = 1 , NELMNT
      AXF(J) = 0.0
      N1 = NODELM(1,J)
      N2 = NODELM(2,J)
      I = J
      NK = NNODE
      TX1 =( X(N2) - X(N1) )
      TY1 =( Y(N2) - Y(N1) )
      TZ1 =( Z(N2) - Z(N1) )
      XP = X(NK) - X(N1)
      YP = Y(NK) - Y(N1)
      ZP = Z(NK) - Z(N1)
      AL(I) = SQRT ( TX1*TX1 + TY1*TY1 + TZ1*TZ1 )
      ALO(I) = AL(I)
      AA = TX1*TX1 + TY1*TY1 + TZ1*TZ1
```

```
      AB = TX1*XP  + TY1*YP  + TZ1*ZP
      U1 = AA*XP - AB*TX1
      U2 = AA*YP - AB*TY1
      U3 = AA*ZP - AB*TZ1
      UU = U1*U1 + U2*U2 + U3*U3
      UU = SQRT(UU)
      CX = ( X(N2) - X(N1)  ) / AL(I)
      CY = ( Y(N2) - Y(N1)  ) / AL(I)
      CZ = ( Z(N2) - Z(N1)  ) / AL(I)
      C(1,J) = CX
      C(2,J) = CY
      C(3,J) = CZ
      C(4,J) = U1 / UU
      C(5,J) = U2 / UU
      C(6,J) = U3 / UU
      C(7,J) = C(2,J)*C(6,J) - C(3,J)*C(5,J)
      C(8,J) = C(3,J)*C(4,J) - C(1,J)*c(6,J)
      C(9,J) = C(1,J)*C(5,J) - C(2,J)*C(4,J)
20    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine BOPTIM used for the D.S.A for the beam element.   *
C **.*****************************************************************
      Forcesub   BOPTIM (A,KA)
     & of NP ident ME
      REAL A(1),TEMP2(9000)
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
      COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA.AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      Shared REAL T1(16),T2(16),T3(16)
      Externf SENSNB,NSAFB
      End declarations
      Barrier
      IF(MALY .EQ. 1 .AND. MUNA .EQ. 1 ) THEN
      CALL DIAMTR
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NOEP,NNPE,ICHK,MALY,MUNA,DISPA,SIGMAA,
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K33),A(K34) )
      ENDIF
      End barrier
      IF(MALY .EQ. 1 .AND. MUNA .EQ. 1 ) THEN
C ---- Linear design sensitivity analysis
      IF ( NLFLAG .EQ. 1 ) THEN
      CALL CPUTIM((T1))
      Forcecall SENSNB
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10A) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),
     & A(K15),A(K32A),KA(M7),KA(M5),NDV,SIGMAA,NSC,NTDC,DISPA,A(K34),
     & NDC )
      CALL CPUTIM((T2))
C ---- Nonlinear design sensitivity analysis
      ELSEIF ( NLFLAG .EQ. 2 ) THEN
      CALL CPUTIM((T1))
      Forcecall NSAFB
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10A) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),
     & A(K15),A(K32A),KA(M7),KA(M5),NDV,SIGMAA,NSC,NTDC,DISPA,A(K34),
     & NDC,A(K31) )
      CALL CPUTIM((T2))
      ENDIF
      ENDIF
      Barrier
      CALL     OCORD (IR,IW,NNODE,A(K14),KA(M1),A(K15),NTD,
     & A(K1),A(K2),A(K3),ICHK)
      CALL FINDFB
```

```
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MALY,MUNA,DISPA,SIGMAA,
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K15),A(K14F),A(K33),A(K34),NOEP )
      CALL OLOADB
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),
     & A(K8A) , A(K8B) , A(K8C) , A(K8I1) , A(K8I2) , A(K8I3),
     & A(K10) , A(K11) , A(K20) , KA(M1), KA(MI)   ,
     & KA(M2)  , KA(M2A) , KA(M3) , KA(M4) , KA(M6),A(K16),A(K14),
     & A(K15),A(K10A),A(K17) )
999   CONTINUE
      End barrier
      RETURN
      END
C **.*****************************************************************
C   This subroutine OPTIMT used for the D.S.A for the truss elment.   *
C **.*****************************************************************
      Forcesub OPTIMT (A,KA)
     & of NP ident ME
      REAL A(1),TEMP2(9000)
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
     COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      Shared REAL T1(16),T2(16),T3(16)
      Externf NSAFT
      End declarations
      Barrier
      CALL STRESS( IR,IW,NELMNT,A(K16),A(K7),KA(M2) )
      End barrier
      CALL CPUTIM(T1(ME))
      IF(MALY .EQ. 1 .AND. MUNA .EQ. 1 ) THEN
C --- Linear design sensitivity analysis.
      IF ( NLFLAG .EQ. 1 ) THEN
      CALL
     &       SENS2 (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
     & A(K10A),A(K11),A(K12),A(K13),A(K20),KA(M1),KA(MI),
     & KA(M2),KA(M3),KA(M4),KA(M6),A(K16),A(K15),A(K31),NDV,NDC,NSC,
     & NTDC )
      CALL
     &       SENS1 (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
     & A(K10),A(K11),A(K12),A(K13),A(K20),KA(M1),KA(MI),
     & KA(M2),KA(M3),KA(M4),KA(M6),A(K16),A(K15),A(K31),NDV,A(K32A),
     & KA(M7),KA(M5),AREAL,SIGMAA,NDC,NSC,NTDC,DISPA)
C --- Nonlinear design sensitivity analysis.
      ELSEIF(NLFLAG .EQ. 2 ) THEN
      Forcecall
     &       NSAFT  (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & A(K1),A(K2),A(K3),A(K4),A(K5),A(K6),A(K7),A(K8) ,
     & A(K10A),A(K11),A(K12),A(K13),A(K20),KA(M1),KA(MI),
     & KA(M2),KA(M3),KA(M4),KA(M6),A(K16),A(K15),A(K31),NDV,A(K32A),
     & KA(M7),KA(M5),AREAL,SIGMAA,NDC,NSC,NTDC,DISPA)
      ENDIF
      ENDIF
      CALL CPUTIM(T2(ME))
899   continue
      Barrier
      CALL SENOUT (IR,IW,NNODE,NELMNT,NTD,NDV,A(K32A),NDC,NSC,NTDC)
      CALL     OCORD (IR,IW,NNODE ,A(K14),KA(M1),A(K15),NTD,
     & A(K1),A(K2),A(K3) ,ICHK)
      CALL
     &       FINDFF (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MALY,MUNA,DISPA,SIGMAA,
     & A(K7),A(K7F),A(K15),A(K14F),A(K16),A(K16F) )
      CALL OLOAD  (IR,IW,NOMP,NELMNT,NNPE,A(K4),A(K5),A(K6),A(K7),
     & A(K8),KA(MI),A(K10),A(K11),A(K12),A(K13),A(K1),A(K2),A(K3),
     & KA(M2) ,ICHK ,A(K10A),NUM4,MULT4 ,KA(M2) ,A(K16) ,KA(M1),
     & NNODE,NDPN,NTD,A(K17),NADYA,DELMDA,A(K14),A(K15))
      End barrier
```

```
      RETURN
      END
C **.*****************************************************************
C  This subroutine PBEAM will generate and assemble the element      *
C  stiffness matrix for the BEAM element using node scheme.          *
C **.*****************************************************************
      Forcesub  PBEAM
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM ,PR ,  DM,WEIGHT,
     & A1 ,  A2   , A3  ,  AI1 ,  AI2 , AI3 ,
     & AL   ,C ,BIGK ,ID    ,NODELM ,MID,MEID,NID1,NID2,NJPR1,NJPR2,
     & MAXA,AXF)
     & of NP ident ME
      REAL X(1),Y(1),Z(1),YM(1),PR(1),DM(1),WEIGHT(1)
      REAL A1(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL BIGK(1),AXF(1)
      REAL AL(1),C(9,NELMNT)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID1(MAXJT,NNODE),NJPR1(1)
      INTEGER MAXA(1),ID(6,NNODE),MEID(1),NID2(MAXJT,NNODE),NJPR2(1)
      Private REAL SMLK(12,12),E,V,G,AREA,AIX,AIY,AIZ,ALL,BB
      Private INTEGER MAN,KCONT,JCONT,LM(24),I,J
      End declarations
      Presched do 1 I = 1 , NDPE
      DO 1 J = 1 , NDPE
      SMLK(I,J) = 0.0
1     End presched do
      IADD = 0
C -------------------------------------------------------
      Barrier
      End barrier
      Presched do 21 KCONT = 1 , NNODE
      DO 11 JCONT = 1 , NJPR1(KCONT)
      MAN = NID1(JCONT,KCONT)
      E =    YM( MID( MAN ) )
      V =    PR(MID(MAN))
      G =    E / (2.0 * ( 1.0 + V ) )
      AREA = A1( MEID( MAN ) )
      AIX  = AI1 ( MEID ( MAN ) )
      AIY  = AI2 ( MEID ( MAN ) )
      AIZ  = AI3 ( MEID ( MAN ) )
      ALL  = AL(MAN)
      BB =  AXF(MAN)  /  ALL
      CALL ES3DB1(MAN,NELMNT,NNODE,SMLK,AREA,E,AIX,AIY,AIZ,C,ALL,
     & NTD,G,1,1,ICHK,BB)
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      DO 10 I = 1 , NDPN
      II = LM(I)
      IF(LM(I) .EQ. 0 ) GO TO 10
      DO 20 J = I ,  NDPN
      JJ = LM(J)
      IF(LM(J) .EQ. 0 ) GO TO 20
      IF( II .LE. JJ ) THEN
      BIGK (MAXA(II) + JJ -II) = BIGK( MAXA(II) + JJ -II)+ SMLK(I,J)
      ELSE
      BIGK (MAXA(JJ) + II -JJ) = BIGK( MAXA(JJ) + II -JJ)+ SMLK(I,J)
      ENDIF
20    CONTINUE
10    CONTINUE
      DO 40 I = 1 , NDPN
      II = LM(I)
      IF(LM(I) .EQ. 0 ) GO TO 40
      DO 30 J = NDPN+ 1 , NDPE
      JJ = LM(J)
      IF(LM(J) .EQ. 0 ) GO TO 30
      IF( II .LE. JJ ) THEN
      BIGK (MAXA(II) + JJ -II) = BIGK( MAXA(II) + JJ -II)+ SMLK(I,J)
      ELSE
      BIGK (MAXA(JJ) + II -JJ) = BIGK( MAXA(JJ) + II -JJ)+ SMLK(I,J)
      ENDIF
30    CONTINUE
40    CONTINUE
11    CONTINUE
21    End presched do
C -------------------------------------------------------
      Barrier
      End barrier
      Barrier
      End barrier
      Presched do 23 KCONT = 1 , NNODE
      DO 13 JCONT = 1 , NJPR2(KCONT)
      MAN = NID2(JCONT,KCONT)
      E =    YM( MID( MAN ) )
      V =    PR(MID(MAN))
      G =    E / (2.0 * ( 1.0 + V ) )
      AREA = A1( MEID( MAN ) )
      AIX  = AI1 ( MEID ( MAN ) )
```

```
      AIY  = AI2 ( MEID ( MAN ) )
      AIZ  = AI3 ( MEID ( MAN ) )
      ALL  = AL(MAN)
      BB =  AXF(MAN) /  ALL
      CALL ES3DB2(MAN,NELMNT,NNODE,SMLK,AREA,E,AIX,AIY,AIZ,C,ALL,
     & NTD,G,1,1,ICHK,BB)
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      DO 60 I = NDPN + 1 , NDPE
      II = LM(I)
      IF(LM(I) .EQ. 0 ) GO TO 60
      DO 50 J = I , NDPE
      JJ = LM(J)
      IF(LM(J) .EQ. 0 ) GO TO 50
      IF( II .LE. JJ ) THEN
      BIGK (MAXA(II) + JJ -II) = BIGK( MAXA(II) + JJ -II)+ SMLK(I,J)
      ELSE
      BIGK (MAXA(JJ) + II -JJ) = BIGK( MAXA(JJ) + II -JJ)+ SMLK(I,J)
      ENDIF
50    CONTINUE
60    CONTINUE
13    CONTINUE
23    End presched do
      RETURN
      END
C **.*****************************************************************
C  This subroutine PLOAD is used to read and calcualte the nodal     *
C  forces.                                                           *
C **.*****************************************************************
      SUBROUTINE PLOAD (IR,IW,NNODE,P,ID,NTDOF,NLD,ICHK,DLTP)
      REAL P(NTDOF),PTEMP(6),DLTP(NTDOF)
      INTEGER ID(6,NNODE)
      DO 1 I = 1 , NTDOF
      P(I) = 0.0
1     CONTINUE
      DO 20 K = 1 , NNODE
      READ(IR,1000) N,L,(PTEMP(J),J=1,6)
      IF ( N .EQ. 0  ) GO TO 30
      DO 10 I = 1 , 6
      ITEMP = ID(I,N)
      IF(ITEMP  .NE. 0 ) THEN
      CONT1 = NLD
      P(ITEMP) = PTEMP(I) / CONT1
      DLTP(ITEMP) = P(ITEMP)
      ENDIF
10    CONTINUE
20    CONTINUE
30    CONTINUE
1000  FORMAT(2I5,6F10.4)
      RETURN
      END
C **.*****************************************************************
C  This subroutine READBM will read the beam data.                   *
C **.*****************************************************************
      SUBROUTINE  READBM (IR,IW,NOMP,NELMNT,NNPE,YM,PR,DM,WIGHT,
     & NOEP,A1,A2,A3,SI1,SI2,SI3 , F,X,Y,Z,AL,ALO,C,
     & NODELM,MPN,IEPN,IFEF,IER,NOFEFS,ICHK)
      REAL YM(1) , PR(1), DM(1) , WIGHT(1),C(9,NELMNT)
      REAL  A1(1), A2(1) , A3(1) , SI1(1) , SI2(1) , SI3(1)
      REAL X(1) , Y(1) , Z(1) , AL(1), F(6,1),ALO(1)
      INTEGER  NODELM(2,NELMNT) , MPN(NELMNT)
      INTEGER IEPN(NELMNT) , IFEF( 4 , NELMNT ) , IER ( 2 , NELMNT )
      DO 10 I = 1 , NOMP
      READ(IR,1000) MIN,YM(MIN),PR(I),DM(I),WIGHT(I)
      WRITE(IW,15) MIN,YM(MIN)
10    CONTINUE
      WRITE(IW,*)'     E L M E N T   P R O P E R T Y '
      WRITE(IW,*)
      WRITE(IW,*)' ID   Ax        Ay        Az        Ix
     & Iy       Iz'
      WRITE(IW,*)
      DO 20 I = 1 , NOEP
      READ(IR,1001)NGP,A1(NGP),A2(NGP),A3(NGP),SI1(NGP),SI2(NGP),
     & SI3(NGP)
      WRITE(IW,1001)NGP,A1(NGP),A2(NGP),A3(NGP),SI1(NGP),SI2(NGP),
     & SI3(NGP)
20    CONTINUE
      DO 30 I = 1 , 3
      READ(IR,1002) X1,X2,X3,X4
30    CONTINUE
      DO 40 I = 1 , NOFEFS
      READ(IR,1003) J,F(1,J),F(2,J),F(3,J),ICC,F(4,J),F(5,J),F(6,J)
      WRITE(IW,1003)J,F(1,J),F(2,J),F(3,J),ICC,F(4,J),F(5,J),F(6,J)
40    CONTINUE
      DO 50 I = 1 , NELMNT
      READ(IR,1004) J,NODELM(1,J),NODELM(2,J),NK,MPN(J),IEPN(J),
     & IFEF(1,J),IFEF(2,J),IFEF(3,J),IFEF(4,J),IER(1,J),IER(2,J),K
      N1 = NODELM(1,J)
```

```
      N2 = NODELM(2,J)
      TX =( X(N2) - X(N1) ) **2
      TY =( Y(N2) - Y(N1) ) **2
      TZ =( Z(N2) - Z(N1) ) **2
      XP = X(NK) - X(N2)
      YP = Y(NK) - Y(N2)
      ZP = Z(NK) - Z(N2)
      AL(I) = SQRT ( TX + TY + TZ )
      ALO(I) = AL(I)
      CX = ( X(N2) - X(N1)  ) / AL(I)
      CY = ( Y(N2) - Y(N1)  ) / AL(I)
      CZ = ( Z(N2) - Z(N1)  ) / AL(I)
      CXZ = SQRT( CX**2 + CZ**2 )
      IF( CXZ .EQ. 0.0 ) THEN
      DO 90 IE = 1 , 9
      C(IE,J) = 0.0
90    CONTINUE
      C(2,J) = CY
      C(4,J) = - CY
      C(9,J) = 1.0
      ELSE
      XS = CX * XP + CY * YP + CZ * ZP
      YS = -CX * CY * XP / CXZ + CXZ * YP -
     & CY * CZ *ZP / CXZ
      ZS = - CZ * XP /CXZ + CX *ZP / CXZ
      CONST = SQRT( YS**2 + ZS**2 )
      SALFA = ZS / CONST
      CALFA = YS / CONST
      C(1,J) = CX
      C(2,J) = CY
      C(3,J) = CZ
      C(4,J) = ( - CX * CY * CALFA - CZ * SALFA ) / CXZ
      C(5,J) =   CXZ * CALFA
      C(6,J) = ( - CY * CZ * CALFA + CX * SALFA ) / CXZ
      C(7,J) = (  CX * CY * SALFA - CZ * CALFA ) / CXZ
      C(8,J) = - CXZ * SALFA
      C(9,J) = (  CY * CZ * SALFA + CX * CALFA ) / CXZ
      ENDIF
50    CONTINUE
15    FORMAT(2X,'MATERIAL ID NO = ',I5 /
     & 2X,'YOUNGS CONSTANT =',F10.2  /)
25    FORMAT(4I5)
35    FORMAT(4E10.2)
45    FORMAT(I5,6E10.2)
55    FORMAT(5x,9F7.3)
1000  FORMAT(I5,4E10.2)
1001  FORMAT(I5,6E10.2)
1002  FORMAT(4F10.0)
1003  FORMAT (I5,6F10.0/I5,6F10.0)
1004  FORMAT(10I5,2I6,I8)
      RETURN
      END
C **.*****************************************************************
C  This subroutine READND will read the nodal points.              *
C **.*****************************************************************
      SUBROUTINE READND (IR,IW,NNODE,X,Y,Z,LN,NTDOF,ICHK)
      REAL X(1),Y(1),Z(1)
      INTEGER LN(6,NNODE)
      CHARACTER CORD*1
      DO 10 J = 1 , NNODE
      READ(IR,1000) CORD,I,(LN(K,I),K=1,6),X(I),Y(I),Z(I),KN,TEMP
      WRITE(1)X(I),Y(I),Z(I)
10    CONTINUE
      ICONT = 1
      DO 30 I = 1 , NNODE
      DO 20 J = 1 , 6
      IF( LN(J,I) .EQ. 0 ) THEN
      LN(J,I) = ICONT
      NTDOF = ICONT
      ICONT = ICONT + 1
      ELSE
      LN(J,I) = 0
      ENDIF
20    CONTINUE
30    CONTINUE
1000  FORMAT(A1,I4,6I5,3F10.0,I5,F10.0)
      RETURN
      END
C **.*****************************************************************
C  This subroutine READTS will read the truss data.                *
C **.*****************************************************************
                    S U B R O U T I N E     R E A D T S
(IR,IW,NOMP,NELMNT,NNPE,YM,CTH,DM,AREA,WIGHT,
     & NODELM,AL,ALO,CX,CY,CZ,X,Y,Z,MPN,ICHK)
      REAL YM(1) , AREA(1) ,CX(1),CY(1),CZ(1) , AL(1),ALO(1)
      REAL X(1) , Y(1) , Z(1) ,CTH(1),DM(1),WIGHT(1)
      INTEGER NODELM(NNPE,NELMNT),MPN(1)
      DO 10 I = 1 , NOMP
```

```
      READ(IR,1000) MIN,YM(MIN),CTH(I),DM(I),AREA(MIN),WIGHT(I)
10     CONTINUE
      DO 20 I = 1 , 4
      READ(IR,1001) X1,X2,X3,X4
20     CONTINUE
      DO 30 I = 1 , NELMNT
      READ(IR,1002) J,NODELM(1,J),NODELM(2,J),MPN(J)
      N1 = NODELM(1,J)
      N2 = NODELM(2,J)
      TX =( X(N2) - X(N1) ) **2
      TY =( Y(N2) - Y(N1) ) **2
      TZ =( Z(N2) - Z(N1) ) **2
      AL(I) = SQRT ( TX + TY + TZ )
      ALO(I) = AL(I)
      CX(I) = ( X(N2) - X(N1)  ) / AL(I)
      CY(I) = ( Y(N2) - Y(N1)  ) / AL(I)
      CZ(I) = ( Z(N2) - Z(N1)  ) / AL(I)
30     CONTINUE
15     FORMAT(2X,'MATERIAL ID NO = ',I5 /
     & 2X,'YOUNGS CONSTANT =',F20.2  /
     & 2X,'AREA OF SECTION =',F20.2  /)
25     FORMAT(4I5)
35     FORMAT(4F10.2)
45     FORMAT(I5,6F10.2)
1000   FORMAT(I5,5F10.0)
1001   FORMAT(4F10.0)
1002   FORMAT(4I5,F10.0,I5)
      RETURN
      END
C **.*****************************************************************
C   This subroutine ROOT will find the step size for the BFGS method. *
C **.*****************************************************************
      Forcesub ROOT (A,KA,S) of NP ident ME
      REAL A(1)
      INTEGER KA(1)
      COMMON/ALLI1/K1,K2,K3,K1A,K2A,K3A,NC1,NC2,NC3,NC4
      COMMON/ALLI2/IFLAG,JFLAG,METHOD,LOOPL,ICHK,ISWTCH
      COMMON/ALLI3/NNODE,NELMNT,NELTYP,NLD,NLFLAG,IW,IW1,IR,NTD
      COMMON/ALLI4/NNPE,NDPE,NDPN,MAXJT,NTRMS,JNLD,NOIIT,MAXNIT
      COMMON/ALLI5/M1,MI,M3,M3A,M4,M4A,M5,M6,M7
      COMMON/ALLI6/K14,K14A,K14B,K14C,K14D,K15,K16,K17,K18,K20,KL20
      COMMON/ALLI7/K31,DELMDA,ISOR,EPS1,EPS2
      COMMON/ALLI8/K40,K41,K42,K43,K44
      COMMON/ALLI9/IOPT,K7F,K14F,K16F
      COMMON/TRUSI1/K4,K5,K6,K7,K8,K10,K10A,K11,K12,K13,KL13,NOMP
      COMMON/BEAMI1/K8A,K8B,K8C,K8I1,K8I2,K8I3
      COMMON/BEAMI2/M2,M2A,M2B,M2C,NOFEFS,NOEP
      COMMON/OPTI1/NDV,NDC,NSC,NTDC,MFINDJ,MFINDL,K32,K32A,KL32,K33,K34

COMMON/OPTR1/DISPA,BL,SIGMAA,AREAL,MAXNOP,MAWYA,MUNA,MALY,K35
      COMMON/BF1/MULT4,NUM4
      Shared REAL T1(16),T2(16),T3(16),T4(16),T5(16),T6(16),T7(16)
      Shared REAL TOL,EPS,TIMELT(16),TIMEAM(16),TIMEBS(16)
      Shared REAL TTNOR,TDISP(16),TNEW(16)
      Shared INTEGER JSTOP,MESTOP,LSTOP,NEWIT
      Shared REAL P,P0,P1,Q0,Q1,S0,S1,Q
      Shared INTEGER NSTOP
      Private INTEGER MECONT
      Shared LOGICAL TYPE7
      Externf EVAL
      End declarations
      NEWIT = 2
      EPS = EPS1
      P0 = .10
      P1 = .11
      NSTOP = 0
      S0 = P0
      Barrier
      End barrier
     Forcecall EVAL(A(1),KA(1),A(K14),A(K14A),A(K42),A(K1),A(K2),A(K3),
    & A(K1A),A(K2A),A(K3A),A(K15),S0,Q0)
      S1 = P1
      Barrier
      End barrier
     Forcecall EVAL(A(1),KA(1),A(K14),A(K14A),A(K42),A(K1),A(K2),A(K3),
    & A(K1A),A(K2A),A(K3A),A(K15),S1,Q1)
      Barrier
      End barrier
C ....          ..... DO LOOP STARTS .....
      DO 100 MECONT = 1 , 20
      Barrier
      TOP = P1 - P0
      BOT = Q1 - Q0
      IF(ABS(BOT) .LE. 1.0E-08) GO TO 99
      P = P1 - Q1 * TOP  / BOT
      DAN = ABS(P-P1) / ABS(P1)
      IF(DAN .LT. EPS ) THEN
      NSTOP = 1
```

```
      ELSE
      P0 = P1
      P1 = P
      S = P
      Q0 = Q1
      ENDIF
      GO TO 101
99    NSTOP = 1
101   CONTINUE
      End barrier
      Barrier
      End barrier
      IF(NSTOP .EQ. 1 ) GO TO 199
     Forcecall EVAL(A(1),KA(1),A(K14),A(K14A),A(K42),A(K1),A(K2),A(K3),
     & A(K1A),A(K2A),A(K3A),A(K15),S,Q1)
      Barrier
      End barrier
100   CONTINUE
199   CONTINUE
      Barrier
      End barrier
      Barrier
      S = P
      IF( MECONT .GE. 20 ) S = 1.0
      IF( S .LE. 0.0 ) S = 1.0
      IF( S .GE. 3.0 ) S = 1.0
      WRITE(IW,*)' FINAL VALUE OF S = ',S
      WRITE(IW,*)' NUMBER OF ROOT ITERATIONS = ', MECONT
      End barrier
      RETURN
      END
C **.*****************************************************************
C   This subroutine ROWC used to solve the linear system of equations *
C   [A]{X} = {B}.                                                     *
C   This subroutine was devoloped by:                                   *
C       1. Tarun K. Agarwal.                                          *
C       2. Duc T. Nguyen.                                             *
C       3. Olaf Storaasli.                                           *
C **.*****************************************************************
      Forcesub ROWC(A,B,MAXA,IROWL,ICOLH,NEQ,NEQP1,NTERMS,IFLAG
     +  ,jops,ICHK,NUM4,MESTOP)    of NP ident ME
      REAL A(NTERMS),B(NEQ)
      INTEGER MAXA(NEQP1),IROWL(NEQ),ICOLH(NEQ)
      INTEGER jops
      INTEGER MESTOP
      Private INTEGER I,J,K,L,IM1,IC1,IBOT,ICOL,ICOLP,ITOP,JROW,KM1
      Private INTEGER JM1,JM2,JM3,JM4,JM5,JM6,JM7,JM8,jm9,IDIV,IDIV1
      Private INTEGER JTOP,JBOT,ICOPY,jj1 ,jjrow ,ll
      Private REAL XMULT1,XMULT2,XMULT3,XMULT4,XMULT5,XMULT6,XMULT7,
     +            XMULT8,TEMP,XINV,SUM
      Async REAL   X(10001)
      End declarations
C     ..................................................................
      IF(IFLAG.EQ.1) THEN
        Barrier
        End Barrier
        Barrier
        End barrier
        IF(MESTOP .EQ. 1 ) RETURN
       Presched  DO 9 I = 1, NEQ
        Void X(I)
9      End Presched Do
       jops = 0
      Barrier
        DO 191 LINDA = 1 , NEQ
        IF( A(MAXA(LINDA)) .LE. 0.0 ) THEN
        WRITE(6,*) '-------> THE DIAGONAL IN K = 0.0 <------- '
        MESTOP = 1
        ENDIF
191     CONTINUE
      jops = 0
       A(1) = SQRT(A(1))
       XINV = 1.0/A(1)
CDIR$ IVDEP
       DO 20 K = 1, IROWL(1)
         A(K+1) = XINV * A(K+1)
20     CONTINUE
       jops = jops + irowl(1)+2
       Produce X(1)=a(1)
c      write(*,*) 'first void has been unvoided'
      End Barrier
      IF(MESTOP. EQ. 1 ) RETURN
C.....DECOMPOSED STIFFNESS MATRIX PHASE
      Presched DO 100 I = 2, NEQ
c       TAKES CARE OF ROWS ONE BY ONE
        im1 = maxa(i)
        ic1 = icolh(i)
        ibot = i - 8*( (i-1)/8 )
```

```
      icol = ic1 - ibot + 1
      icolp= icol/8
      itop = icol - 8*icolp
c ....indices calculation for modifcation by itop elements.
      jrow = i - ic1
      jm1  = maxa(jrow) + ic1
      jjrow=irowl(jrow)
      IF (ITOP. GE. 1 ) THEN
         ICOPY = JROW + ITOP -1
C********         If (Isfull(x(icopy))) go to 331
         Copy X(ICOPY) INTO TEMP
      ENDIF
331    continue
      go to (101,102,103,104,105,106,107,108), itop

C.....
      go to 150
CDIR$ IVDEP
 101    do 111 k = 1, jjrow-ic1+1
        km1 = k -1
        a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
 111    continue
      go to 150

 102    jm2 = jm1 + jjrow
CDIR$ IVDEP
      do 112 k = 1, jjrow-ic1+1
        km1 = k -1
        a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +             - a(jm2)*a(jm2+km1)
 112    continue
      go to 150
 103    jm2 = jm1 + jjrow
      jm3 = jm2 + jjrow -1
CDIR$ IVDEP
      do 113 k = 1, jjrow -ic1+1
        km1 = k -1
        a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
     +             -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
 113    continue
      go to 150
 104    jm2 = jm1 + jjrow
      jm3 = jm2 + jjrow -1
      jm4 = jm3 + jjrow -2
CDIR$ IVDEP
      do 114 k = 1, jjrow -ic1+1
        km1 = k -1
        a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
     +             -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +             -a(jm4)*a(jm4+km1)
 114    continue
      go to 150
 105    jm2 = jm1 + jjrow
      jm3 = jm2 + jjrow -1
      jm4 = jm3 + jjrow -2
      jm5 = jm4 + jjrow -3
CDIR$ IVDEP
      do 115 k = 1, jjrow -ic1+1
        km1 = k -1
        a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
     +             -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +             -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
 115    continue
      go to 150
 106    jm2 = jm1 + jjrow
      jm3 = jm2 + jjrow -1
      jm4 = jm3 + jjrow -2
      jm5 = jm4 + jjrow -3
      jm6 = jm5 + jjrow -4
CDIR$ IVDEP
      do 116 k = 1, jjrow -ic1+1
        km1= k -1
        a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +             -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +             -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
     +             -a(jm6)*a(jm6+km1)
 116    continue
      go to 150
 107    jm2 = jm1 + jjrow
      jm3 = jm2 + jjrow -1
      jm4 = jm3 + jjrow -2
      jm5 = jm4 + jjrow -3
      jm6 = jm5 + jjrow -4
      jm7 = jm6 + jjrow -5
CDIR$ IVDEP
      do 117 k = 1, jjrow -ic1+1
        km1 = k -1
        a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
```

```
     +                -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
     +                -a(jm6)*a(jm6+km1) -a(jm7)*a(jm7+km1)
 117     continue
         go to 150
 108     continue
CDIR$ IVDEP
C.....................................................................
 150     jops = jops + itop*(jjrow -ic1+2)*2
         ll =  1
         idiv =  1
         if (icolp.le.ll) then
           ll =icolp
           idiv1=1
         else
           idiv1=icolp-ll+1
         endif
         jtop = ic1
         jbot = ic1-itop+1
         do 10 l = 1,  ll
           jtop = jtop - itop
           jbot = jbot - 8*idiv1
           itop = 8*idiv1
           idiv1 = idiv
            if (l.eq.ll) then
           icopy = i - 1
           else
           icopy = i -jbot +ibot-1
             endif
c*******        If (Isfull(x(icopy))) go to 332
         Copy X(icopy) into temp

332       continue
          do 200 j = jtop, jbot, -8
             JJ1 = I-J
             jjrow = irowl(jj1)
             jm1 = maxa(jj1) + j
             jm2 = jm1 + jjrow
             jm3 = jm2 + jjrow -1
             jm4 = jm3 + jjrow -2
             jm5 = jm4 + jjrow -3
             jm6 = jm5 + jjrow -4
             jm7 = jm6 + jjrow -5
             jm8 = jm7 + jjrow -6
             xmult1 = a(jm1)
CDIR$ IVDEP
          DO 300 K = 1, jjrow -J +1
             KM1 = K -1
             A(im1+km1) = A(im1+km1)
   .                -a(jm1)*a(jm1+km1)  -a(jm2)*a(jm2+km1)
   .                -a(jm3)*a(jm3+km1)  -a(jm4)*a(jm4+km1)
   .                -a(jm5)*a(jm5+km1)  -a(jm6)*a(jm6+km1)
   .                -a(jm7)*a(jm7+km1)  -a(jm8)*a(jm8+km1)
 300       CONTINUE
          jops = jops + 16*( jjrow -j+1)
 200      CONTINUE
 10      continue
          ll=i-1
          Copy x(ll) into temp
 333      continue
          go to (201,202,203,204,205,206,207,208) ibot-1
            go to 250
c.....................................................................
 201        jjrow = irowl(i-1)
           jm1 = maxa(i-1) +1
CDIR$ IVDEP
           DO 211 K = 1, jjrow
             KM1 = K -1
             A(IM1+KM1) = A(IM1+KM1) - a(jm1)* A(JM1 +KM1)
 211        CONTINUE
          go to 250
 202         jjrow = irowl(i-2)
           jm1 = maxa(i-2) +2
           JM2 = jm1 + jjrow
CDIR$ IVDEP
           DO 212 K = 1, jjrow -1
             KM1 = K -1
             A(IM1+KM1) = A(IM1+KM1) - a(jm1)*a(jm1+km1)
   .                -A(jm2)*A(JM2+KM1)
 212         CONTINUE
          go to 250

 203         jjrow = irowl(i-3)
           jm1 = maxa(i-3) + 3
           JM2 = jm1 + jjrow
           JM3 = jm2 + jjrow -1
CDIR$ IVDEP
           DO 213 K = 1, jjrow -2
```

```
            KM1=K -1
            A(IM1+KM1) = A(IM1+KM1) -A(jm1)*A(JM1+KM1)
     .             -a(jm2)*A(JM2+KM1)-a(jm3)*A(JM3+KM1)
 213        CONTINUE
      go to 250
 204        jjrow = irowl(i-4)
            jm1 = maxa(i-4) + 4
            jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
CDIR$ IVDEP
            do 214 k = 1,jjrow  -3
              km1 = k -1
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .             -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .             -a(jm4)*a(jm4+km1)
 214        continue
      go to 250
 205        jjrow = irowl(i-5)
            jm1 = maxa(i-5) + 5
            jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
CDIR$ IVDEP
            do 215 k = 1, jjrow -4
              km1 = k -1
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .             -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .             -a(jm4)*a(jm4+km1)-a(jm5)*A(jm5+km1)
 215        continue
      go to 250
 206        jjrow = irowl(i-6)
            jm1 = maxa(i-6) +6
            jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
            jm6 = jm5 + jjrow -4
CDIR$ IVDEP
            do 216 k = 1, jjrow -5
              km1 = k -1
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .             -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .             -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     .             -a(jm6)*a(jm6+km1)
 216        continue
      go to 250
 207        jjrow = irowl(i-7)
            jm1 = maxa(i-7)+7
            jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
            jm6 = jm5 + jjrow -4
            jm7 = jm6 + jjrow -5
CDIR$ IVDEP
            do 217 k = 1, jjrow -6
              km1 = k -1
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .             -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .             -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     .             -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
 217        continue
      go to 250
 208        continue
CDIR$ IVDEP
C.....

 250      jops = jops + 2*(ibot-1)*(jjrow -ibot +2)
          A(IM1) =SQRT(A(IM1))
          XINV = 1.0/A(IM1)
CDIR$ IVDEP
          DO 260  K = 1, IROWL(I)
            A(IM1+K) = XINV *A(IM1+K)
 260      CONTINUE
          jops = jops + irowl(i) +2
          Produce X(I) = A(IM1)
 100      End Presched Do
          ELSE
C.....FORWARD REDUCTION

          Barrier
            CCMAX = 0.0
            DO 278 IYT = 1 , NEQ
            CCMAX = MAX(CCMAX,A(MAXA(IYT)) )
 278        CONTINUE
```

```
      DET = 1.0
      DO 877 IYT=1,NEQ
      IF(DET .LE. 1.0E-20) THEN
      GO TO 739
      ENDIF
      DET = ( A(MAXA(IYT)) / CCMAX ) * DET
877    CONTINUE

      DET = DET * DET
739    CONTINUE
      jops = 0
     DO 510 I = 1,NEQ
      B(I) = B(I)/A(MAXA(I))
      SUM = B(I)
      IM1 =MAXA(I)
CDIR$ IVDEP
      DO 520 J = I+1, I+IROWL(I)
        B(J) = B(J) - SUM* A(IM1+J-I)
520    CONTINUE
      jops = jops + 2*(irowl(i))+ 2
510   CONTINUE


C.....BACK SUBSTITUTION
     B(NEQ) = B(NEQ)/A(MAXA(NEQ))
     jops = jops +1
     DO 1010 I = NEQ-1,1,-1
       SUM = 0.0
CDIR$ IVDEP
      DO 1020 J =I+1, IROWL(I)+I
        SUM=SUM+ A(MAXA(I)+J-I)*B(J)
1020    CONTINUE
      B(I) =(B(I)-SUM)/A(MAXA(I))
          jops = jops + 2*(irowl(i)) +2
1010  CONTINUE
    End Barrier
    ENDIF
    RETURN
    END
C **.*****************************************************************
C  This subroutine ROWLNG is used to find the row length of each row *
C  in the total stiffness matrix.                                     *
C **.*****************************************************************
     SUBROUTINEROWLNG(IR,IW,NEQ,IROWL,MAXA,NTERMS,NNODE,NELMNT,
    & LOOPL,NDPE,ID,ICOLH,NODELM,NNPE,ICHK)
     INTEGER   MAXA(1),IROWL(1),ID(6,NNODE),ICOLH(1),LM(24)
     INTEGER   NODELM(NNPE,NELMNT)
     NDPN = NDPE / NNPE
     DO 10 I = 1, NEQ
10    IROWL(I) = 0
     DO 40 J = 1 , NELMNT
     CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,J,
    & NDPN )
     MAXD = 0
     DO 20 I = 1 , NDPE
     II = LM(I)
     IF ( II .GT. MAXD ) MAXD = II
20    CONTINUE
     DO 30 I = 1 , NDPE
     II = LM(I)
     IF(II .EQ. 0 ) GO TO 30
     IRWTMP = MAXD - II + 1
     IF ( IRWTMP .GT. IROWL(II) )  IROWL(II) = IRWTMP
30    CONTINUE
40    CONTINUE
     NBLOK = NEQ / LOOPL
     LFTOVR = NEQ - (NBOLK * LOOPL )
     MAXCOL = 0
     DO 70 I = 1 , NBLOK
     ISTART = ( I-1 ) * LOOPL + 1
     IEND = I * LOOPL
     DO 50 JROW = ISTART , IEND
     JCOL = JROW + IROWL ( JROW ) - 1
     IF ( JCOL .GT. MAXCOL ) MAXCOL = JCOL
50    CONTINUE
     DO 60 JROW = ISTART , IEND
     IROWL(JROW) = MAXCOL - JROW + 1
60    CONTINUE
70    CONTINUE
     ISTART = NBLOK * LOOPL + 1
     IEND = NEQ
     DO 80 JROW = ISTART , IEND
80    IROWL(JROW) = NEQ - JROW + 1
     MAXA(1) = 1
     DO 90 I = 2 , NEQ
     MAXA(I) = MAXA(I-1) + IROWL(I-1)
90    CONTINUE
     NBAND = 0
```

```
      DO 91 I = 1 , NEQ
      NBAND = MAX(IROWL(I) , NBAND )
91     IROWL(I) = IROWL(I) - 1
      WRITE(6,*)' THE MAX BANDWIDTH =     ',NBAND
      NTERMS = MAXA(NEQ)
      RETURN
      END
C **.*****************************************************************
C  This subroutine SENOUT will write out the sensitivty answers for  *
C  each design variable.                                             *
C **.*****************************************************************
      SUBROUTINE SENOUT
     &            (IR,IW,NNODE,NELMNT,NTD,NDV,DG,NDC,NSC,NTDC)
      REAL DG(NDV,NTDC)
      IW1 = 9
      WRITE(IW1,*)' The Outout in Subroutine SENOUT '
      IF( NDC .LE. 10 ) THEN
      NENDI = NDC
      NENDJ = NTDC
      ELSE
      NENDI = 1
      NENDJ = NDC+1
      ENDIF
      DO 30 II = 1 , NDV
      WRITE(IW1,25) II
      WRITE(IW1 ,*)
      WRITE(IW1 ,*)' DEGREE OF FREEDOM          SENSITIVITY'
      WRITE(IW1 ,*)'====================          ==========='
      DO 10 JJ = 1 , NENDI
      WRITE( IW1,15) JJ,DG(II,JJ)
10     CONTINUE
      WRITE(IW1 ,*)
      WRITE(IW1 ,*)' MEMBER NO:                 SENSITIVITY'
      WRITE(IW1 ,*)'====================          ==========='
      DO 20 JJ = NDC+1 ,NENDJ
      WRITE(IW1 ,15) JJ,DG(II,JJ)
20     CONTINUE
30     CONTINUE
15     FORMAT(I10,10X,F15.6)
25     FORMAT(/26X,'DESIGN VARIABLE NO:',I5/25X,'----------------------'
     & //)
      RETURN
      END
C **.*****************************************************************
C  This subroutine SEN1 is used to find the DSA for the truss element*
C  using linear analysis.                                            *
C **.*****************************************************************
      SUBROUTINE SENS1
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM,  CF,  DM,  AR,WEIGHT,
     & AL  ,CX  ,CY  ,CZ  ,BIGK ,ID   ,NODELM   ,
     & MID  ,NID  ,NJPROW,MAXA,AXF,TOTLD,DKDB,NDV,DG,IROWL,ICOLH,BL,
     & SIGMAA,NDC,NSC,NTDC,DEFA)
      REAL  X(1),Y(1),Z(1),YM(1),CF(1),DM(1),AR(1),WEIGHT(1)
      REAL  BIGK(1),AXF(1),TOTLD(1),DG(NDV,NTDC),TEMP1(100),DD(6)
      REAL  AL(1),CX(1),CY(1),CZ(1),DKDB(NDV,NTD),CCX(3)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24),ICOLH(1),IROWL(1)
C-----------------------------------------------------------------*
      WRITE(9 ,*)
      WRITE(9,*)' SENSITIVITY ANALYSIS USING AJOINT METHOD IN SENS1'
      NDEFC = NDEFC + NDV
20     CONTINUE
      ICONT = NDV
      ICONT = 0
C ****  This will find the derivatevs with respect to the Displacement
      DO 60 I     = 1,NDC
      ICONT = ICONT + 1
      DO 30 J = 1 , NTD
      IF(I .EQ. J ) THEN
      TEMP1(J) = 2.0* TOTLD(I) / DEFA
      ELSE
      TEMP1(J) = 0.0
      ENDIF
30     CONTINUE
      NEQP1 = NTD + 1
      CALL
     &      SOLVER (BIGK  ,TEMP1 ,MAXA  ,IROWL ,ICOLH ,NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK)
      DO 50 M = 1 , NDV
      SUM = 0.0
      DO 40 J = 1 , NTD
      SUM = SUM + DKDB(M,J) * TEMP1(J)
40     CONTINUE
      DG(M,ICONT) =  SUM
50     CONTINUE
60     CONTINUE
```

```
C *** THIS WILL FIND THE DERIVITAVES WITH RESPECT TO THE STRESSES
99    CONTINUE
      DO 140 MAN  = 1 , NELMNT
      DO 118 II = 1 , NTD
118   TEMP1(II) = 0.0
      ICONT = ICONT + 1
      CCX(1) = CX(MAN)
      CCX(2) = CY(MAN)
      CCX(3) = CZ(MAN)
      E =    YM( MID( MAN ) )
      ALL = AL(MAN)
      AREA = AR ( MID(MAN) )
      CALL      SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      DO 70 J = 1 , 6
      ICONST = LM(J)
      IF(ICONST .EQ. 0 )THEN
      DD(J) = 0
      ELSE
      DD(J) = TOTLD(ICONST)
      ENDIF
70    CONTINUE
      DO 80 NANY = 1 , 3
      II = LM(NANY)
      JJ = LM(NANY+3)
      IF( II .NE. 0 ) THEN
      TEMP1(II) = - CCX(NANY) * E / (  ALL * SIGMAA )
      ENDIF
      IF( JJ .NE. 0 ) THEN
      TEMP1(JJ) =  CCX(NANY) * E /(   ALL * SIGMAA )
      ENDIF
80    CONTINUE
      CALL
     &      SOLVER (BIGK  ,TEMP1 ,MAXA  ,IROWL ,ICOLH ,NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK)
      DO 100 M = 1 , NDV
      SUM = 0.0
      DO 90 J = 1 , NTD
      SUM = SUM + DKDB(M,J) * TEMP1(J)
90    CONTINUE
      DG(M,ICONT) = -SUM
100   CONTINUE
140   CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine SENS2 is used to find the derivitave of the      *
C  tangent stiffness matrix with respect to the design variable.    *
C **.*****************************************************************
      SUBROUTINE SENS2
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM,  CF,  DM,  AR,WEIGHT,
     & AL  ,CX  ,CY  ,CZ  ,BIGK ,ID   ,NODELM   ,
     & MID  ,NID  ,NJPROW,MAXA,AXF,TOTLD,DKDB,NDV,NDC,NSC,NTDC)
      REAL  X(1),Y(1),Z(1),YM(1),CF(1),DM(1),AR(1),WEIGHT(1)
      REAL  BIGK(1),AXF(1),TOTLD(1)
      REAL  AL(1),CX(1),CY(1),CZ(1),SMLK(6,6),DKDB(NDV,NTD)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24)
      DO 1 I = 1 , NDPE
      DO 1 J = 1 , NDPE
1     SMLK(I,J) = 0.0
      IADD = 0
      DO 2 I = 1 , NDV
      DO 2 J = 1 , NTD
      DKDB(I,J) = 0.0
2     CONTINUE
C -------------------------------------------------------
      DO 11 MAN = 1 , NELMNT
      E =    YM( MID( MAN ) )
      BB = 0.0
      AREA = 1.00
      JIM = MID(MAN)
      CALL ES3DT0(MAN,NELMNT,NNODE,SMLK,AREA,E,AMOI,CX,CY,CZ,AL,NTD,
     & 1,4,BB)
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      DO 10 I = 1 , NDPE
      II = LM(I)
      IF(LM(I) .EQ. 0 ) GO TO 10
      DO 20 J = 1 ,  NDPE
      JJ = LM(J)
      IF(LM(J) .EQ. 0 ) GO TO 20
      DKDB(JIM,II) = DKDB(JIM,II) + SMLK(I,J) * TOTLD(JJ)
20    CONTINUE
10    CONTINUE
11    CONTINUE
```

```
21    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine SENSAJ will find the sensitivity of truss elemnt  *
C   with respect to the area.                                         *
C **.*****************************************************************
      Forcesub SENSAJ
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM,  CF,  DM,  AR,WEIGHT,
     & AL  ,CX  ,CY  ,CZ  ,BIGK ,ID   ,NODELM   ,
     & MID  ,NID  ,NJPROW,MAXA,AXF,TOTLD,DKDB,NDV,DG,IROWL,ICOLH,BL,
     & SIGMAA,NDC,NSC,NTDC,DEFA)
     & of NP ident ME
      REAL  X(1),Y(1),Z(1),YM(1),CF(1),DM(1),AR(1),WEIGHT(1),CCX(3)
      REAL  BIGK(1),AXF(1),TOTLD(1),DG(NDV,NTDC)
      REAL  AL(1),CX(1),CY(1),CZ(1),DKDB(NDV,NTD),DCOL(3)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24),ICOLH(1),IROWL(1)
      Private REAL TEMP1(9000),TEMP2(9000)
      Shared REAL T1(16),T2(16),T3(16)
      End declarations
C-----------------------------------------------------------------*
      CALL CPUTIM(T1(ME))
      Barrier
      WRITE(9 ,*)
      WRITE(9,*)' SENSITIVITY ANALYSIS USING AJOINT METHOD IN SENSAJ'
      ICONT = 0
      End barrier
      Presched do 80 NADYA = 1 , NDV
      DO 10 II = 1 , NTD
      TEMP2(II) = 0.0
10    CONTINUE
      DO 30 MAN = 1 , NELMNT
      IF( NADYA .EQ.  MID( MAN ) ) THEN
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK2,MAN,
     & NDPN )
      J = MAN
      PP = AXF(MAN)
      DCOL(1) = CX(J)
      DCOL(2) = CY(J)
      DCOL(3) = CZ(J)
      DO 20 II = 1 , 3
      JJ = LM(II)
      MM = LM(II+3)
      IF(JJ . NE. 0 ) THEN
      TEMP2(JJ) = TEMP2(JJ) + PP * DCOL(II)
      DKDB(NADYA,JJ) = TEMP2(JJ)
      ENDIF
      IF( MM .NE. 0 ) THEN
      TEMP2(MM) = TEMP2(MM) - PP * DCOL(II)
      DKDB(NADYA,MM) = TEMP2(MM)
      ENDIF
20    CONTINUE
      ENDIF
30    CONTINUE
      ICONT = ICONT + 1
      DO 60 I     = 1,NTD
      DO 40 J = 1 , NTD
      IF(I .EQ. J ) THEN
      TEMP1(J) = 2.0* TOTLD(I) / DEFA
      ELSE
      TEMP1(J) = 0.0
      ENDIF
40    CONTINUE
      NEQP1 = NTD + 1
      CALL
     &      SOLVER (BIGK  ,TEMP1 ,MAXA  ,IROWL ,ICOLH ,NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK,1,1)
      SUM = 0.0
      DO 50 J = 1 , NTD
      SUM = SUM + TEMP2(J) * TEMP1(J)
50    CONTINUE
      DG(NADYA,I) =  SUM
60    CONTINUE
80    End presched do
C *** THIS WILL FIND THE DERIVITAVES WITH RESPECT TO THE STRESSES
      CALL CPUTIM(T2(ME))
      Barrier
      TMAX = 0.0
      DO 300  I = 1 , NP
      TMAX = MAX(  TMAX,( T2(I) - T1(I)) )
300   CONTINUE
      WRITE(IW,*) ' MAX TIME FOR SENSITIVITY ANALYSIS = ',TMAX
      ICONT = NDC
      Presched do 140 MCONT  = 1 , NDV
      ICONT = NDC
```

```
      DO 118 II = 1 , NTD
      TEMP2(II) = DG(MCONT,II)  / (2.0 * TOTLD(II) / DEFA )
118   CONTINUE
      DO 148 MAN = 1 , NELMNT
      ICONT = ICONT + 1
      CCX(1) = CX(MAN)
      CCX(2) = CY(MAN)
      CCX(3) = CZ(MAN)
      E =    YM( MID( MAN ) )
      ALL = AL(MAN)
      AREA = AR ( MID(MAN) )
      CALL      SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      SUM = 0.0
      DO 180 NANY = 1 , 3
      II = LM(NANY)
      JJ = LM(NANY+3)
      IF( II .NE. 0 ) THEN
      SUM = SUM  + TEMP2(II) *  CCX(NANY) * E / ( ALL * SIGMAA )
      ENDIF
      IF( JJ .NE. 0 ) THEN
      SUM = SUM  - TEMP2(JJ) *  CCX(NANY) * E /( ALL * SIGMAA )
      ENDIF
180   CONTINUE
      DG(MCONT,ICONT) = - SUM
148   CONTINUE
100   CONTINUE
140   End presched do
      RETURN
      END
C **.*****************************************************************
C   This subroutine SENSNB used to calculate the DSA for the beam     *
C   element.                                                          *
C **.*****************************************************************
      Forcesub   SENSNB
     &           (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM ,PR ,  DM,WEIGHT,
     & A1 ,  A2   , A3 ,  AI1 ,  AI2 , AI3 ,
     & AL  ,C ,BIGK, ID    ,NODELM ,MID,MEID,NID,NJPROW,MAXA,AXF,
     & TOTLD,DG,IROWL,ICOLH,NDV,SIGMAA,NSC,NTDC,DEFA,DOUT,NDC )
     & of NP ident ME
      REAL X(1),Y(1),Z(1),YM(1),PR(1),DM(1),WEIGHT(1)
      REAL A1(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL AXF(1),TOTLD(1),DG(NDV,NTDC),TEMP1(7000)
      REAL AL(1),C(9,NELMNT),DOUT(1),BIGK(1)
      REAL TEMP2(7000),D2(7000),D3(7000)
      REAL CCX(10),CCY(10),CCZ(10),CX(10),CY(10),CZ(10)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24),MEID(1),ICOLH(1),IROWL(1)
      Private INTEGER ICONT
      End declarations
      IWF = 10
      Barrier
      WRITE(IWF,*)
      WRITE(IWF,*)' SENSITIVITY ANALYSIS USING AJOINT METHOD '
      CALL STOPP ( IR,IW,NNODE,NELMNT)
      End barrier
      ICONT = 0
      PI = ACOS( -1.0 )
      Presched do 80 nadya = 1 , ndv
      do 77 ii = 1 , ntd
      temp2(ii) = 0.0
77    continue
      DO 11 MAN = 1 , NELMNT
      IF( NADYA .EQ.  MEID (MAN) ) THEN
      I = MAN
      AREA = .50 * PI * DOUT( MEID( I ) )
      E    = YM( MID( I ) )
      AIX  = PI * ( (DOUT (MEID(I)) ) **3 ) / 8.00
      AIY  = PI * ( (DOUT (MEID(I)) ) **3 ) / 16.00
      AIZ  = PI * ( (DOUT (MEID(I)) ) **3 ) / 16.00
      V    = PR( MID(I) )
      G    = E / (2.0 * (1 + V ) )
      ALL = AL( MAN )
      CALL SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,I,
     & NDPN )
      DO 51 J = 1 , 12
      II = LM(J)
      IF(II .NE. 0 ) THEN
      D2(J) = TOTLD(II)
      ELSE
      D2(J) = 0.0
      ENDIF
51    CONTINUE
      CALL ES3DBS(I,NELMNT,NNODE,AREA,E,AIX,AIY,AIZ,C,ALL,NTD,
     & G,ICHK,D3,D2,AXF)
      DO 40 II = 1 , 6
```

```
      JJ = LM(II)
      MM = LM(II+6)
      IF(JJ . NE. 0 ) THEN
      TEMP2(JJ) = TEMP2(JJ) +  D3(II)
      ENDIF
      IF( MM .NE. 0 ) THEN
      TEMP2(MM) = TEMP2(MM) + D3(II+6)
      ENDIF
40    CONTINUE

      ENDIF
11    continue
      ICONT = ICONT + 1
      DO 60 I     = 1,NTD
      DO 30 J = 1 , NTD
      TEMP1(J) = 0.0
30    CONTINUE
      TEMP1(I) = 1.0 / DEFA
      NEQP1 = NTD + 1
      CALL
     &       SOLVER (BIGK  ,TEMP1 ,MAXA  ,IROWL ,ICOLH ,NTD,NEQP1,
     & NTRMS,2,JOPS,ICHK,0,0    )
      SUM = 0.0
      DO 46 J = 1 , NTD
      SUM = SUM + TEMP2(J) * TEMP1(J)
46    CONTINUE
      DG(NADYA,I) = - SUM
50    CONTINUE
60    CONTINUE
80    continue
C *** THIS WILL FIND THE DERIVITAVES WITH RESPECT TO THE STRESSES
99    CONTINUE
      ICONT = NDC
      MAJDI2 = 1
      IF(MAJDI2 .EQ. 1 ) GO TO 998
      DO 140 MCONT  = 1 , NDV
      ICONT = NDC
      DO 118 II = 1 , NTD
      TEMP2(II) = DG(MCONT,II)  / (2.0 * TOTLD(II) / DEFA )
118   CONTINUE
      DO 148 MAN = 1 , NELMNT
      ICONT = ICONT + 1
      CCX(1) = CX(MAN)
      CCX(2) = CY(MAN)
      CCX(3) = CZ(MAN)
      E =    YM( MID( MAN ) )
      ALL = AL(MAN)
      CALL       SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      SUM = 0.0
      DO 180 NANY = 1 , 3
      II = LM(NANY)
      JJ = LM(NANY+3)
      IF( II .NE. 0 ) THEN
      SUM = SUM  + TEMP2(II) *  CCX(NANY) * E / ( ALL * SIGMAA )
      ENDIF
      IF( JJ .NE. 0 ) THEN
      SUM = SUM  - TEMP2(JJ) *  CCX(NANY) * E /( ALL * SIGMAA )
      ENDIF
180   CONTINUE
      DG(MCONT,ICONT) = - SUM
148   CONTINUE
100   CONTINUE
140   CONTINUE
888   continue
998   CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine ROWC used to solve the linear system of equations *
C  [A]{X} = {B}.(sequenial)                                          *
C  This subroutine was devoloped by:                                 *
C       1. Tarun K. Agarwal.                                         *
C       2. Duc T. Nguyen.                                            *
C       3. Olaf Storaasli.                                           *
C **.*****************************************************************
      SUBROUTINE SOLVER (A,B,MAXA,IROWL,ICOLH,NEQ,NEQP1,
     & NTERMS,IFLAG,JOPS,ICHK,NUM4,MESTOP)

      REAL A(NTERMS),B(NEQ)
      INTEGER MAXA(NEQP1),IROWL(NEQ),ICOLH(NEQ)
      INTEGER jops
      INTEGER MESTOP
      INTEGER I,J,K,L,IM1,IC1,IBOT,ICOL,ICOLP,ITOP,JROW,KM1
      INTEGER JM1,JM2,JM3,JM4,JM5,JM6,JM7,JM8,IDIV,IDIV1
      INTEGER JTOP,JBOT,ICOPY,JJ1 ,JJROW ,LL
      REAL XMULT1,XINV,SUM
      REAL   X(10001)
```

```
C     .....

      IF(IFLAG.EQ.1) THEN
       MESTOP = 0
       jops = 0
        DO 191 LINDA = 1 , NEQ
        IF( A(MAXA(LINDA)) .LE. 0.0 ) THEN
        WRITE(6,*) '--------> THE DIAGONAL IN K = 0.0 <------- '
        MESTOP = 1
        ENDIF
        IF(MESTOP .EQ. 1 ) RETURN
191      CONTINUE
       jops = 0
       A(1) = SQRT(A(1))
       XINV = 1.0/A(1)
CDIR$ IVDEP
       DO 20 K = 1, IROWL(1)
         A(K+1) = XINV * A(K+1)
20     CONTINUE
       jops = jops + irowl(1)+2
            X(1)=A(1)
C.....DECOMPOSED STIFFNESS MATRIX PHASE

            DO 100 I = 2, NEQ

       im1 = maxa(i)
       ic1 = icolh(i)
       ibot = i - 8*( (i-1)/8 )
       icol = ic1 - ibot + 1
       icolp= icol/8
       itop = icol - 8*icolp
       jrow = i - ic1
       jm1  = maxa(jrow) + ic1
       jjrow=irowl(jrow)
       IF (ITOP. GE. 1 ) THEN
          ICOPY = JROW + ITOP -1
       ENDIF
331     continue
       go to (101,102,103,104,105,106,107,108), itop
C.....
       go to 150

CDIR$ IVDEP
101      do 111 k = 1, jjrow-ic1+1
          km1 = k -1
          a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
111      continue
       go to 150

102      jm2 = jm1 + jjrow
CDIR$ IVDEP
         do 112 k = 1, jjrow-ic1+1
          km1 = k -1
          a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +                - a(jm2)*a(jm2+km1)
112      continue
       go to 150

103      jm2 = jm1 + jjrow
         jm3 = jm2 + jjrow -1
CDIR$ IVDEP
         do 113 k = 1, jjrow -ic1+1
          km1 = k -1
          a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
     +                -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
113      continue
       go to 150

104      jm2 = jm1 + jjrow
         jm3 = jm2 + jjrow -1
         jm4 = jm3 + jjrow -2
CDIR$ IVDEP
         do 114 k = 1, jjrow -ic1+1
          km1 = k -1
          a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
     +                -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                -a(jm4)*a(jm4+km1)
114      continue
       go to 150

105      jm2 = jm1 + jjrow
         jm3 = jm2 + jjrow -1
         jm4 = jm3 + jjrow -2
         jm5 = jm4 + jjrow -3
CDIR$ IVDEP
         do 115 k = 1, jjrow -ic1+1
          km1 = k -1
          a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
```

```
     +                -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
 115    continue
        go to 150

 106    jm2 = jm1 + jjrow
        jm3 = jm2 + jjrow -1
        jm4 = jm3 + jjrow -2
        jm5 = jm4 + jjrow -3
        jm6 = jm5 + jjrow -4
CDIR$ IVDEP
        do 116 k = 1, jjrow -ic1+1
          km1= k -1
          a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +                -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
     +                -a(jm6)*a(jm6+km1)
 116    continue
        go to 150

 107    jm2 = jm1 + jjrow
        jm3 = jm2 + jjrow -1
        jm4 = jm3 + jjrow -2
        jm5 = jm4 + jjrow -3
        jm6 = jm5 + jjrow -4
        jm7 = jm6 + jjrow -5
CDIR$ IVDEP
        do 117 k = 1, jjrow -ic1+1
          km1 = k -1
          a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +                -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
     +                -a(jm6)*a(jm6+km1) -a(jm7)*a(jm7+km1)
 117    continue
        go to 150
 108    continue
CDIR$ IVDEP
C.....
 150    jops = jops + itop*(jjrow -ic1+2)*2
        ll =  2
        idiv =  1
        if (icolp.le.ll) then
          ll =icolp
          idiv1=1
        else
          idiv1=icolp-ll+1
        endif
        jtop = ic1
        jbot = ic1-itop+1

        do 10 l = 1,  ll
          jtop = jtop - itop
          jbot = jbot - 8*idiv1
          itop = 8*idiv1
          idiv1 = idiv

          if (l.eq.ll) then
          icopy = i - 1
          else
          icopy = i -jbot +ibot-1
           endif
 332     continue
         do 200 j = jtop, jbot, -8
            JJ1 = I-J
            jjrow = irowl(jj1)
            jm1 = maxa(jj1) + j
            jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
            jm6 = jm5 + jjrow -4
            jm7 = jm6 + jjrow -5
            jm8 = jm7 + jjrow -6
            xmult1 = a(jm1)
CDIR$ IVDEP
         DO 300 K = 1, jjrow -J +1
            KM1 = K -1
            A(im1+km1) = A(im1+km1)
     .              -a(jm1)*a(jm1+km1)  -a(jm2)*a(jm2+km1)
     .              -a(jm3)*a(jm3+km1)  -a(jm4)*a(jm4+km1)
     .              -a(jm5)*a(jm5+km1)  -a(jm6)*a(jm6+km1)
     .              -a(jm7)*a(jm7+km1)  -a(jm8)*a(jm8+km1)
 300      CONTINUE
         jops = jops + 16*( jjrow -j+1)
 200     CONTINUE
 10     continue

        ll=i-1
```

```
333       continue
          go to (201,202,203,204,205,206,207,208) ibot-1
            go to 250
c.....
 201        jjrow = irowl(i-1)
           jm1 = maxa(i-1) +1
CDIRS IVDEP
           DO 211 K = 1, jjrow
             KM1 = K -1
             A(IM1+KM1) = A(IM1+KM1) - a(jm1)* A(JM1 +KM1)
 211        CONTINUE
          go to 250
 202        jjrow = irowl(i-2)
           jm1 = maxa(i-2) +2
           JM2 = jm1 + jjrow
CDIRS IVDEP
           DO 212 K = 1, jjrow -1
             KM1 = K -1
             A(IM1+KM1) = A(IM1+KM1) - a(jm1)*a(jm1+km1)
     .                -A(jm2)*A(JM2+KM1)
 212         CONTINUE
          go to 250

 203        jjrow = irowl(i-3)
           jm1 = maxa(i-3) + 3
           JM2 = jm1 + jjrow
           JM3 = jm2 + jjrow -1
CDIRS IVDEP
           DO 213 K = 1, jjrow -2
             KM1=K -1
             A(IM1+KM1) = A(IM1+KM1) -A(jm1)*A(JM1+KM1)
     .                -a(jm2)*A(JM2+KM1)-a(jm3)*A(JM3+KM1)
 213         CONTINUE
         go to 250
 204        jjrow = irowl(i-4)
           jm1 = maxa(i-4) + 4
           jm2 = jm1 + jjrow
           jm3 = jm2 + jjrow -1
           jm4 = jm3 + jjrow -2
CDIRS IVDEP
           do 214 k = 1,jjrow -3
             km1 = k -1
             a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .                -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .                -a(jm4)*a(jm4+km1)
 214         continue
          go to 250
 205        jjrow = irowl(i-5)
           jm1 = maxa(i-5) + 5
           jm2 = jm1 + jjrow
           jm3 = jm2 + jjrow -1
           jm4 = jm3 + jjrow -2
           jm5 = jm4 + jjrow -3
CDIRS IVDEP
           do 215 k = 1, jjrow -4
             km1 = k -1
             a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .                -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .                -a(jm4)*a(jm4+km1)-a(jm5)*A(jm5+km1)
 215         continue
          go to 250
 206        jjrow = irowl(i-6)
           jm1 = maxa(i-6) +6
           jm2 = jm1 + jjrow
           jm3 = jm2 + jjrow -1
           jm4 = jm3 + jjrow -2
           jm5 = jm4 + jjrow -3
           jm6 = jm5 + jjrow -4
CDIRS IVDEP
           do 216 k = 1, jjrow -5
             km1 = k -1
             a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .                -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .                -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     .                -a(jm6)*a(jm6+km1)
 216         continue
          go to 250
 207        jjrow = irowl(i-7)
           jm1 = maxa(i-7)+7
           jm2 = jm1 + jjrow
           jm3 = jm2 + jjrow -1
           jm4 = jm3 + jjrow -2
           jm5 = jm4 + jjrow -3
           jm6 = jm5 + jjrow -4
           jm7 = jm6 + jjrow -5
CDIRS IVDEP
           do 217 k = 1, jjrow -6
             km1 = k -1
```

```
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     .                   -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .                   -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     .                   -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
 217           continue
           go to 250

 208         continue
CDIR$ IVDEP
C.....

 250       jops = jops + 2*(ibot-1)*(jjrow -ibot +2)
           A(IM1) =SQRT(A(IM1))
           XINV = 1.0/A(IM1)
CDIR$ IVDEP
           DO 260  K = 1, IROWL(I)
             A(IM1+K) = XINV *A(IM1+K)
 260       CONTINUE
           jops = jops + irowl(i) +2
                X(I) = A(IM1)
 100       continue
           ELSE
C.....FORWARD REDUCTION
           CCMAX = 0.0
           DO 278 IYT = 1 , NEQ
           CCMAX = MAX(CCMAX,A(MAXA(IYT)) )
278         CONTINUE

           DET = 1.0
           DO 877 IYT=1,NEQ
           IF(DET .LE. 1.0E-20) THEN
           GO TO 739
           ENDIF
           DET = ( A(MAXA(IYT)) / CCMAX ) * DET
877         CONTINUE
           DET = DET * DET
739         CONTINUE
           jops = 0
          DO 510 I = 1,NEQ
            B(I) = B(I)/A(MAXA(I))
            SUM = B(I)
            IM1 =MAXA(I)
CDIR$ IVDEP
           DO 520 J = I+1, I+IROWL(I)
              B(J) = B(J) - SUM* A(IM1+J-I)
520         CONTINUE
           jops = jops + 2*(irowl(i))+ 2
510       CONTINUE


C.....BACK SUBSTITUTION
          B(NEQ) = B(NEQ)/A(MAXA(NEQ))
          jops = jops +1
          DO 1010 I = NEQ-1,1,-1
            SUM = 0.0
CDIR$ IVDEP
           DO 1020 J =I+1, IROWL(I)+I
             SUM=SUM+ A(MAXA(I)+J-I)*B(J)
1020       CONTINUE
           B(I) =( B(I) - SUM ) / A( MAXA(I) )
         jops = jops + 2*(irowl(i)) +2
1010    CONTINUE
       ENDIF
       RETURN
       END
C **.*****************************************************************
C  This subroutine PVCONJ is the S.O.R. method used for solving the  *
C  linear system of equations [A]{x}={b}.                            *
C **.*****************************************************************
        Forcesub   PVCONJ(IR,IW,N,NTERMS,A,B,X,MAXNIT,EPSLON,
       & IROWL,MAXA,ICHK,P,R ,T,AA ,ICOLH )
       & of NP ident ME
        REAL A(1),X(1),B(1),P(1),R(1),T(1),AA(1)
        Shared REAL RPTMP(3000),RTMP(3000),RP(3000)
        INTEGER MAXA(1),ICOLH(1),IROWL(1)
        Shared INTEGER NP1,ICHANG
        Shared REAL    TOL2,ALFA2,ALFA4,ALFA,BETA,IADD2,IADD,ALFA3
        Shared  REAL  PALF4(16),PALF2(16),PALF3(16)
        End declarations
        Barrier
        TOL2 = 1.0E-08
        NP1 = N+1
        ICHANG = 0
C ***************************************************************
C ...  ICHANG .EQ. 0 --> Use [L][U] as a Precondition Matrix      *
C ...  ICHANG .EQ. 1 --> Use [I] as a Precondition Matrix        *
C ...  ICHANG .EQ. 2 --> Use Jacobian Precondion (Diagonal of [A])  *
C ***************************************************************
```

```
      IF(ICHANG .EQ. 1 ) THEN
      DO 3 I = 1 , N
      X(I) = 1.0
3     CONTINUE
      ELSEIF( ICHANG .EQ. 2 ) THEN
      DO 4 I = 1 , NTERMS
      AA(I ) = 0.0
4     CONTINUE
      DO  6 I = 1 , N
      AA(MAXA(I)) =  A(MAXA(I))
      AA(MAXA(I)) =  1.0
6     CONTINUE
      ENDIF
      ICHK2 = 0
      IF(ICHK2 .EQ. 1 ) THEN
      WRITE(IW,*)' ..... S U B R O U T I N E  C O N J .....'
      WRITE(6,*)' THE A MATRIX IS = '
      WRITE(IW,15)(A(I),I=1,NTERMS)
      WRITE(6,*)' THE AA MATRIX IS = '
      WRITE(IW,15)(AA(I),I=1,NTERMS)
      WRITE(6,*)' THE B VECTOR IS = '
      WRITE(IW,15)(B(I) , I=1,N)
      WRITE(6,*)' THE X VECTOR IS = '
      WRITE(IW,15)(X(I) , I=1,N)
      WRITE(IW,*)' MAXA(I) '
      WRITE(IW,25)(MAXA(I) , I = 1 ,N)
      WRITE(IW,*)' IROWL(I)'
      WRITE(IW,25)(IROWL(I) , I = 1 , N)
      ENDIF
      End barrier
      IADD = 0
      CALL CPUTIM(T3)
      Forcecall PVATX  (IR,IW,N,A,X,R,MAXA,IROWL,ICOLH,IADD2)
      IADD = IADD + IADD2
      Barrier
      End barrier
      Presched do  7 I = 1 , N
      R(I) = B(I) - R(I)
      RP(I) = R(I)
7     End presched do
      Barrier
      End barrier
      Forcecall ROWC(AA,RP,MAXA,IROWL,ICOLH,N,NP1,NTERMS,2
     +  ,jops,ICHK,NUM4,MESTOP)
      Barrier
      ALFA2 = 0.0
      DO 10 I = 1 , N
      P(I) = RP(I)
      ALFA2 = ALFA2 + RP(I)*R(I)
C     WRITE(6,*)' P(',I,') = ',P(I)
10    CONTINUE
      End barrier
      MAXNIT =100
C ***************************************************************
C      D O  L O O P  O F  C O N J  S T A R T S       *
C ***************************************************************
      DO 100 ICONT = 1 , MAXNIT
C ...> Locate (ALFA)                          3.4.9a
      IADD2 = 0
      IQQ = 0
C ---           [A]*{P} = {T}
      CALLCPUTIM(T1)
      Forcecall PVATX  (IR,IW,N,A,P,T,MAXA,IROWL,ICOLH,IADD2)
      CALL CPUTIM(T2)
      ALFA3  = 0.0
      PALF3(ME) = 0.0
      Presched do 20 I = 1 , N
      PALF3(ME) = PALF3(ME) + P(I) * T(I)
20    End presched do
      Barrier
      DO 102 II = 1 , NP
      ALFA3 = PALF3(II) + ALFA3
102   CONTINUE
      ALFA = - ALFA2 / ALFA3
      End barrier
C ...> Find the new value of {X}                  3.4.9.b
      Presched do 30 I = 1 , N
      X(I) = X(I) - ALFA * P(I)
30    End presched do
      Barrier
      End barrier
C ...< Find the new value of {X}                  3.4.9.b
      IF(ALFA2 .LT. TOL2 ) GO TO 110

C ...> Find the new value of {r} = {r} + Alfa * {T}  3.4.9.c
      PALF2(ME) = 0.0
      ALFA4  = 0.0
      PALF4(ME) = 0.0
```

```
      Presched do 40 I = 1 , N
      RTMP(I) = R(I)
      R(I) = R(I) + ALFA * T(I)
      RPTMP(I) = RP(I)
      PALF4(ME) = PALF4(ME) + (RPTMP(I) * RTMP(I))
      RP(I) = R(I)
40    End presched do
      Barrier
      DO 101 II = 1 , NP
      ALFA4 = PALF4(II) + ALFA4
      ALFA2 = 0.0
101   CONTINUE
      End barrier
C ...< Find the new value of {r} = {r} + Alfa * {T}  3.4.9.c
C ...> Solve [M]{rp} = {r}                          3.4.9.d
      JOPS = 0
      Forcecall ROWC(AA,RP,MAXA,IROWL,ICOLH,N,NP1,NTERMS,2
     +  ,JOPS,ICHK,NUM4,MESTOP)
C     IADD = IADD + JOPS
C ...< Solve [M]{rp} = {r}                          3.4.9.d
C ...> Calculate Beta  = (rp,r)/(rptmp,rtmp)        3.4.9.f
C     IADD3 = N*2 + 1
      Presched do 88 I = 1 , N
      PALF2(ME) = PALF2(ME) + (RP(I)*R(I))
88    End presched do
      Barrier
      Do 103 II = 1 , NP
      ALFA2 = PALF2(II) + ALFA2
103   CONTINUE
      BETA = ALFA2/ALFA4
      End barrier
C     IADD = IADD + 2 * N + 1
C ...< Calculate Beta  = (rp,r)/(rptmp,rtmp)        3.4.9.f
C ...> Calculate new {P} = pr + Beta * {P}          3.4.9.g
C     IADD = IADD + N * 2
C     IADD3 = N*2
      Presched do 50 I = 1 , N
      P(I) = RP(I) + BETA * P(I)
50    End presched do
C ...< Calculate new {P} = pr + Beta * {P}          3.4.9.g
      Barrier
      End barrier
100   CONTINUE
C ***************************************************************
C     DO LOOP OF CONJ ENDS !!!!!    *
C ***************************************************************
110   CONTINUE
      CALL CPUTIM(T4)
      Barrier
      WRITE(IW,*)' NUMBER OF ITERATIONS IN CONJ = ',ICONT
      IF(ICHK .EQ. 1 ) THEN
      ENDIF
      End barrier
      Presched do 130 I = 1 , N
      B(I) = X(I)
130   End presched do
      Barrier
      End barrier
15    FORMAT(5E14.2/5E12.2/5E12.2)
25    FORMAT (5I10)
      RETURN
      END
C **.***********************************************************
C  This subroutine PVSOR is the S.O.R. method used for solving the  *
C  linear system of equations [A]{x}={b}.                           *
C **.***********************************************************
      Forcesub PVSOR (IR,IW,N,NTERMS,A,B,X,MAXNIT,EPSLON,
     & IROWL,MAXA,ICHK,C,X1,X2 ,W) of NP ident ME
      REAL A(1),X(1),B(1),C(1),X1(1),X2(1)
      INTEGER IROWL(1),MAXA(1)
      Private REAL    CP(10000),XT(10000)
      Shared REAL R(10000)
      Shared REAL W1,SUM4
      Private REAL SUM2,SUM3,SUM,SUM8
      Private INTEGER IJEND,II,MM,J,I,ICOLH(1)
      Shared LOGICAL TYPE1,TYPE2,TYPE3
      Externf PVATX
      End declarations
      W = 1.00
      Barrier
      DO 3 I = 1 , N
      XT(I) = X(I)
3     CONTINUE
      W = 1.0000
      write(iw,*)' OMEGA = ',W
      W1 = 1.0 - W
      End barrier
      TOL = 1.0 * EPSLON
```

```
      TOL2 = 1.0E-4
      TOL2 = 15.0000
      MAXNIT = 400
C ***************************************************************
C ************* MAJOR DO LOOP STARTES**********
C ***************************************************************
      DO 40 ICONT = 1 , MAXNIT
      SUM2 = 0.0
      SUM4 = 0.0
      DO 1 IJI = 1 , N
      C(IJI) = 0.0
      CP(IJI) = 0.0
1     CONTINUE
      Presched do 30 I = 1 , N - 1
      IJEND = I + IROWL(I)
      MM = MAXA(I) - I
      DO 20 J = I+1 , IJEND
      II = MM + J
      CP(J) = CP(J) +A(II) * X(I)
20    CONTINUE
30    End presched do
      Barrier
      End barrier
      Critical TYPE1
      DO 86 I = 1 , N
      C(I) = C(I) + CP(I)
86    CONTINUE
      End critical
      Barrier
      X(N) = W * (B(N) - C(N)) /A(MAXA(N)) + (W1 * X(N) )
      End barrier
      Presched do 31 I = N-1 , 1 , -1
      IJEND = I + IROWL(I)
      SUM = B(I) - C(I)
      MM = MAXA(I) - I
      DO 10 J = I+1 , IJEND
      II = MM + J
      SUM = SUM - A(II) * X(J)
10    CONTINUE
      X(I) = W * SUM / A(MAXA(I)) + (W1 * X(I) )
31    End presched do
      Barrier
      End barrier
      SUM4 = 0.0
      Forcecall PVATX  (IR,IW,N,A,X,R,MAXA,IROWL,ICOLH,IADD2)
      SUM8 = 0.0
      Presched do 97 I = 1 , N
      SUM8 = SUM8 +(( B(I) -  R(I))**2)
97    End presched do
      Critical TYPE2
      SUM4 = SUM4 + SUM8
      End critical
      Barrier
      End barrier
      SUM2 = SQRT(SUM4)
      IF( SUM2 .LE. TOL2 ) GO TO 49
      Barrier
      End barrier
40    CONTINUE
49    CONTINUE
      Barrier
50    WRITE(IW,*)' NUMBER OF GUSS-SIDEL ITERATION = ',ICONT
      DO 60 I = 1 , N
      B(I) = X(I)
60    CONTINUE
      End barrier
      RETURN
      END
C **.***********************************************************
C  This subroutine STRESS will change the axial loads to stresses   *
C **.***********************************************************
      SUBROUTINE STRESS(IR,IW,NELMNT,AXF,AR,MID)
      REAL  AXF(1),AR(1)
      INTEGER MID(1)
      DO 10 I = 1 , NELMNT
      AXF(I) = AXF(I) / AR(MID(I))
10    CONTINUE
      RETURN
      END
C **.***********************************************************
C  This subroutine SUBLM is used to find the D.O.F associalte with   *
C  each element.                                                     *
C **.***********************************************************
       SUBROUTINE SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,I,
     & NDPN )
       INTEGER ID(6,NNODE) ,NODELM(NNPE,NELMNT),LM(24)
       ICONT = 1
       DO 20 LINDA = 1 , NNPE
```

```
      JJ = NODELM ( LINDA , I )
      DO 10 J = 1 , NDPN
      LM(ICONT) = ID(J,JJ)
      ICONT = ICONT + 1
10    CONTINUE
20    CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine TRANS will transpose the local elemnet stifness   *
C  matrix. (6X6)                                                      *
C **.*****************************************************************
      SUBROUTINE TRANS ( NE,T , C , J ,ICHK )
      REAL  C(9,NE) , T(6,6)
      DO 20 NN = 1 , 6
      DO 20 MM = 1 , 6
      T(NN,MM) = 0.0
20    CONTINUE
      NCONT = 0
      DO 30 NN = 1 , 3
      DO 30 MM = 1 , 3
      NCONT = NCONT + 1
      T(NN,MM) = C(NCONT,J)
30    CONTINUE
      NCONT = 0
      DO 40 NN = 4 , 6
      DO 40 MM = 4 , 6
      NCONT = NCONT + 1
      T(NN,MM) = C(NCONT,J)
40    CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine TRANS1  will transpose the local elemnet stifness *
C  matrix. (12X12)                                                    *
C **.*****************************************************************
      SUBROUTINE TRANS1 ( NE,T , C , J ,ICHK,NSM )
      REAL  C(9,NE) , T(12,12)
      DO 20 NN = 1 , 12
      DO 20 MM = 1 , 12
      T(NN,MM) = 0.0
20    CONTINUE
      NCONT = 0
      DO 30 NN = 1 , 3
      DO 30 MM = 1 , 3
      NCONT = NCONT + 1
      T(NN,MM) = C(NCONT,J)
30    CONTINUE
      NCONT = 0
      DO 40 NN = 4 , 6
      DO 40 MM = 4 , 6
      NCONT = NCONT + 1
      T(NN,MM) = C(NCONT,J)
40    CONTINUE
      NCONT = 0
      DO 50 NN = 7 , 9
      DO 50 MM = 7 , 9
      NCONT = NCONT + 1
      T(NN,MM) = C(NCONT,J)
50    CONTINUE
      NCONT = 0
      DO 60 NN = 10 , 12
      DO 60 MM = 10 , 12
      NCONT = NCONT + 1
      T(NN,MM) = C(NCONT,J)
60    CONTINUE
      RETURN
      END
C **.*****************************************************************
C  This subroutine TRANS1  will transpose the local elemnet stifness *
C  matrix. (12X12)                                                    *
C **.*****************************************************************
      SUBROUTINE TRANS2 ( NE,T , C , J ,ICHK,NSM )
      REAL  C(9,NE) , T(12,12)

      DO 20 NN = 1 , 12
      DO 20 MM = 1 , 12
      T(NN,MM) = 0.0
20    CONTINUE
      NCONT = 0
      DO 30 NN = 1 , 3
      DO 30 MM = 1 , 3
      NCONT = NCONT + 1
      T(MM,NN) = C(NCONT,J)
30    CONTINUE
      NCONT = 0
      DO 40 NN = 4 , 6
      DO 40 MM = 4 , 6
```

```
      NCONT = NCONT + 1
      T(MM,NN) = C(NCONT,J)
40    CONTINUE
      NCONT = 0
      DO 50 NN = 7 , 9
      DO 50 MM = 7 , 9
      NCONT = NCONT + 1
      T(MM,NN) = C(NCONT,J)
50    CONTINUE
      NCONT = 0
      DO 60 NN = 10 , 12
      DO 60 MM = 10 , 12
      NCONT = NCONT + 1
      T(MM,NN) = C(NCONT,J)
60    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine TRANS will transpose the local elemnet stifness  *
C   matrix. (6X6)                                                     *
C **.*****************************************************************
      SUBROUTINE TRANST ( NE , T , C , J ,ICHK )
      REAL  C(9,NE) , T(6,6)
      DO 20 NN = 1 , 6
      DO 20 MM = 1 , 6
      T(NN,MM) = 0.0
20    CONTINUE
      NCONT = 0
      DO 30 NN = 1 , 3
      DO 30 MM = 1 , 3
      NCONT = NCONT + 1
      T(MM,NN) = C(NCONT , J)
30    CONTINUE
      NCONT = 0
      DO 40 NN = 4 , 6
      DO 40 MM = 4 , 6
      NCONT = NCONT + 1
      T(MM,NN) = C(NCONT , J)
40    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine TRUSS used to assemble the element stiffness     *
C   matrix for the truss element.                                     *
C **.*****************************************************************
      Forcesub TRUSS
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,
     & X   ,Y   ,Z   ,  YM,  CF,  DM,  AR,WEIGHT,
     & AL  ,CX  ,CY  ,CZ  ,BIGK ,ID   ,NODELM   ,
     & MID  ,NID1,NID2,NJPR1,NJPR2,MAXA,AXF    ) of NP ident ME
      REAL  X(1),Y(1),Z(1),YM(1),CF(1),DM(1),AR(1),WEIGHT(1)
      REAL  BIGK(1),AXF(1)
      REAL  AL(1),CX(1),CY(1),CZ(1)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID1(MAXJT,NNODE),NJPR1(1)
      INTEGER MAXA(1),ID(6,NNODE),NID2(MAXJT,NNODE),NJPR2(1)
      Private INTEGER MAN,KCONT,JCONT,LM(24),I,J
      Private REAL AREA,E,BB,SMLK(6,6)
      End declarations
      DO 1 I = 1 , NDPE
      DO 1 J = 1 , NDPE
1     SMLK(I,J) = 0.0
      IADD = 0
C -------------------------------------------------------
      Barrier
      End barrier
      Presched do 21 KCONT = 1 , NNODE
      DO 11 JCONT = 1 , NJPR1(KCONT)
      MAN = NID1(JCONT,KCONT)
      E =    YM( MID( MAN ) )
      AREA = AR( MID( MAN ) )
      BB = AXF ( MAN ) / AL ( MAN )
      CALL   ES3DT1(MAN,NELMNT,NNODE,SMLK,AREA,E,AMOI,CX,CY,CZ,AL,NTD,1,1
     & ,BB)
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      DO 10 I = 1 , NDPN
      II = LM(I)
      IF(LM(I) .EQ. 0 ) GO TO 10
      DO 20 J = I ,  NDPN
      JJ = LM(J)
      IF(LM(J) .EQ. 0 ) GO TO 20
      IF( II .LE. JJ ) THEN
      BIGK (MAXA(II) + JJ -II) = BIGK( MAXA(II) + JJ -II)+ SMLK(I,J)
      ELSE
      BIGK (MAXA(JJ) + II -JJ) = BIGK( MAXA(JJ) + II -JJ)+ SMLK(I,J)
      ENDIF
20    CONTINUE
```

```
10    CONTINUE
11    CONTINUE
21    End presched do
C -------------------------------------------------------------
      Presched do 22 KCONT = 1 , NNODE
      DO 12 JCONT = 1 , NJPR1(KCONT)
      MAN = NID1(JCONT,KCONT)
      E =    YM( MID( MAN ) )
      AREA = AR( MID( MAN ) )
      BB = AXF ( MAN ) / AL ( MAN )
                                                C       A       L       L
ES3DT2(MAN,NELMNT,NNODE,SMLK,AREA,E,AMOI,CX,CY,CZ,AL,NTD,1,4
     & ,BB)
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      DO 40 I = 1 , NDPN
      II = LM(I)
      IF(LM(I) .EQ. 0 ) GO TO 40
      DO 30 J = NDPN+ 1 , NDPE
      JJ = LM(J)
      IF(LM(J) .EQ. 0 ) GO TO 30
      IF( II .LE. JJ ) THEN
      BIGK (MAXA(II) + JJ -II) = BIGK( MAXA(II) + JJ -II)+ SMLK(I,J)
      ELSE
      BIGK (MAXA(JJ) + II -JJ) = BIGK( MAXA(JJ) + II -JJ)+ SMLK(I,J)
      ENDIF
30    CONTINUE
40    CONTINUE
12    CONTINUE
22    End presched do
C -------------------------------------------------------------
      Barrier
      End Barrier
      Presched do  23 KCONT = 1 , NNODE
      DO 13 JCONT = 1 , NJPR2(KCONT)
      MAN = NID2(JCONT,KCONT)
      E =    YM( MID( MAN ) )
      AREA = AR( MID( MAN ) )
      BB = AXF ( MAN ) / AL ( MAN )
      CALLES3DT4(MAN,NELMNT,NNODE,SMLK,AREA,E,AMOI,CX,CY,CZ,AL,NTD,4,
     & 4,BB)
      CALL  SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,MAN,
     & NDPN )
      DO 60 I = NDPN + 1 , NDPE
      II = LM(I)
      IF(LM(I) .EQ. 0 ) GO TO 60
      DO 50 J = I , NDPE
      JJ = LM(J)
      IF(LM(J) .EQ. 0 ) GO TO 50
      IF( II .LE. JJ ) THEN
      BIGK (MAXA(II) + JJ -II) = BIGK( MAXA(II) + JJ -II)+ SMLK(I,J)
      ELSE
      BIGK (MAXA(JJ) + II -JJ) = BIGK( MAXA(JJ) + II -JJ)+ SMLK(I,J)
      ENDIF
50    CONTINUE
60    CONTINUE
13    CONTINUE
23    End presched do
      Barrier
      End Barrier
      RETURN
      END
C **.*****************************************************************
C  This subroutine UBFIB is used to find the unbalnced forces in     *
C  the beam element.                                                 *
C **.*****************************************************************
      SUBROUTINE UBFIB
     &          (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MULT4,
     & X   ,Y   ,Z   ,  YM ,PR ,  DM,WEIGHT,TOTLD,
     & AR ,  A2   , A3 ,  AI1 ,  AI2 , AI3 ,
     & AL   ,C ,BIGK ,ID    ,NODELM ,MID,MEID,NID,NJPROW,MAXA,AXF,
     & P,DELP,TOL,JSTOP,METHOD,NUM4,ALO)
      REAL X(1),Y(1),Z(1),YM(1),PR(1),DM(1),WEIGHT(1)
      REAL AR(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL BIGK(1),AXF(1),D3(12),P(1),DELP(1),D2(12)
      REAL AL(1),C(9,NELMNT),TOTLD(1),ALO(1)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),LM(24),MEID(1)
      CONST1 = MULT4
      WRITE(IW,*) ' CONST1 ',CONST1
      do 20 J = 1 , NTD
      P(J) = DELP(J)  + DELP(J) * CONST1
      P(J) = 0.0
20    CONTINUE
      DO 30 I = 1 , NELMNT
      AREA = AR( MEID( I ) )
      E    = YM( MID( I ) )
```

```
      AIX  = AI1 ( MEID(I) )
      AIY  = AI2 ( MEID(I) )
      AIZ  = AI3 ( MEID(I) )
      V    = PR( MID(I) )
      G    = E / (2.0 * (1 + V ) )
      ALL = ALO(I)
      N1 = NODELM(1,I)
      N2 = NODELM(2,I)
      NK = NNODE
      IF(ICHK .EQ. 1 ) THEN
      WRITE(6,*)' NK = ',NK
      WRITE(IW,*)' AREA = ',AREA
      WRITE(IW,*)' E = ',E
      WRITE(IW,*)' AIX = ',AIX
      WRITE(IW,*)' AIY = ',AIY
      WRITE(IW,*)' AIZ = ',AIZ
      ENDIF
      TX1 =( X(N2) - X(N1) )
      TY1 =( Y(N2) - Y(N1) )
      TZ1 =( Z(N2) - Z(N1) )
      XP = X(NK) - X(N1)
      YP = Y(NK) - Y(N1)
      ZP = Z(NK) - Z(N1)
      AL(I) = SQRT ( TX1*TX1 + TY1*TY1 + TZ1*TZ1 )
      ALL = AL(I)
      AA = TX1*TX1 + TY1*TY1 + TZ1*TZ1
      AB = TX1*XP  + TY1*YP  + TZ1*ZP
      U1 = AA*XP - AB*TX1
      U2 = AA*YP - AB*TY1
      U3 = AA*ZP - AB*TZ1
      UU = U1*U1 + U2*U2 + U3*U3
      UU = SQRT(UU)
      CX = ( X(N2) - X(N1)  ) / AL(I)
      CY = ( Y(N2) - Y(N1)  ) / AL(I)
      CZ = ( Z(N2) - Z(N1)  ) / AL(I)
      C(1,I) = CX
      C(2,I) = CY
      C(3,I) = CZ
      C(4,I) = U1 / UU
      C(5,I) = U2 / UU
      C(6,I) = U3 / UU
      C(7,I) = C(2,I)*C(6,I) - C(3,I)*C(5,I)
      C(8,I) = C(3,I)*C(4,I) - C(1,I)*C(6,I)
      C(9,I) = C(1,I)*C(5,I) - C(2,I)*C(4,I)
      CALL SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,I,
     & NDPN )
      DO 51 J = 1 , 12
      II = LM(J)
      IF(II .NE. 0 ) THEN
      D2(J) = TOTLD(II)
      ELSE
      D2(J) = 0.0
      ENDIF
   51 CONTINUE
      CALL ES3DBN(I,NELMNT,NNODE,AREA,E,AIX,AIY,AIZ,C,ALL,
     & NTD,G,ICHK,D3,D2,AXF,IW)
      DO 40 II = 1 , 6
      JJ = LM(II)
      MM = LM(II+6)
      IF(JJ . NE. 0 ) THEN
      P(JJ) = P(JJ) +  D3(II)
      ENDIF
      IF( MM .NE. 0 ) THEN
      P(MM) = P(MM) + D3(II+6)
      ENDIF
   40 CONTINUE
      UFNORM = 0.0
      DO 62 II = 1 , NTD
      UFNORM = UFNORM + P(II) * P(II)
      IF(UFNORM .GT. 1.0E20) GO TO 92
   62 CONTINUE
   92 UFNORM = SQRT(UFNORM)
      WRITE(IW,*)' UNBALANCED FORCE NORM = ',UFNORM
   30 CONTINUE
      IF( NNODE .LE.  5 ) THEN
      WRITE(IW,*)'              ==================== '
      WRITE(IW,*)'               TOTAL  U. B. F '
      WRITE(IW,*)'              ==================== '
      DO 72 II = 1 , NTD
      WRITE(IW,*)' P(',II,') = ',P(II)
   72 CONTINUE
      ENDIF
      DO 31 JJ = 1 , NTD
      P(JJ) = DELP(JJ)  + DELP(JJ) * CONST1 - P(JJ)
   31 CONTINUE
      IF( NNODE .LE. 10 ) THEN
      WRITE(IW,*)'                 ================= '
      WRITE(IW,*)'                    U. B. F '
```

```
      WRITE(IW,*)'                ================ '
      DO 71 II = 1 , NTD
      WRITE(IW,*)' P(',II,') = ',P(II)
71    CONTINUE
      ENDIF
      IF( METHOD .GT. -1 ) THEN
      UFNORM = 0.0
      DO 60 II = 1 , NTD
      UFNORM = UFNORM + P(II) * P(II)
      IF(UFNORM .GT. 1.0E20) GO TO 91
60    CONTINUE
91    UFNORM = SQRT(UFNORM)
      WRITE(IW,*)' UNBALANCED FORCE NORM = ',UFNORM
      IF( UFNORM .LE. TOL ) THEN
      DO 80 J = 1 , NTD
      P(J) = DELP(J)  + P(J)
80    CONTINUE
      JSTOP = 1
      ENDIF
      WRITE(IW,*)' UFNORM = ',UFNORM,'  TOL = ', TOL
      ENDIF
      RETURN
      END
C **.*****************************************************************
C   This subroutine UBFIB2 is used to find the unbalance forces for   *
C   the beam element.                                                 *
C **.*****************************************************************
      Forcesub UBFIB2
     &            (IR,IW,NNODE,NELMNT,NTD,NDPE,NDPN,JFLAG,MAXJT,
     & NTRMS,NOMP,NNPE,ICHK,MULT4,
     & X    ,Y    ,Z    ,  YM ,PR  ,  DM,WEIGHT,TOTLD,
     & AR ,  A2   , A3  ,  AI1 ,  AI2 , AI3 ,
     & AL   ,C ,BIGK ,ID     ,NODELM ,MID,MEID,NID,NJPROW,MAXA,AXF,
     & P,DELP,TOL,JSTOP,METHOD,NUM4,ALO)
     & of NP ident ME
      REAL X(1),Y(1),Z(1),YM(1),PR(1),DM(1),WEIGHT(1)
      REAL AR(1)  , A2(1) ,  A3(1) , AI1(1) , AI2(1) , AI3(1)
      REAL BIGK(1),AXF(1),P(1),DELP(1)
      REAL AL(1),C(9,NELMNT),TOTLD(1),ALO(1)
      INTEGER NODELM(NNPE,NELMNT),MID(1),NID(MAXJT,NNODE),NJPROW(1)
      INTEGER MAXA(1),ID(6,NNODE),MEID(1)
      Private REAL AREA,E,AIX,AIY,AIZ,V,G,ALL ,CONST1
      Private REAL TX1,TY1,TZ1,XP,YP,ZP,AA,AB,U1,U2,U3,UU,CX,CY,CZ
      Private INTEGER N1,N2,JJ,MM,I
      Private REAL PPRV(2000),D2(12),D3(12)
      Private INTEGER LM(24)
      Shared LOGICAL TYPE20
      Shared LOGICAL TYPE11
      Shared LOGICAL TYPE12
      End declarations
      Barrier
      End barrier
      Critical TYPE11
      DO 11 II = 1 , NTD
      PPRV(II) = 0.0
11    CONTINUE
      End critical
      CONST1 = MULT4
      Barrier
      End barrier
      Presched do 20 J = 1 , NTD
      P(J) = DELP(J)  + DELP(J) * CONST1
20    End presched do
      Barrier
      End barrier
      Presched do 30 I = 1 , NELMNT
      AREA = AR( MEID( I ) )
      E    = YM( MID( I ) )
      AIX  = AI1 ( MEID(I) )
      AIY  = AI2 ( MEID(I) )
      AIZ  = AI3 ( MEID(I) )
      V    = PR( MID(I) )
      G    = E / (2.0 * (1 + V ) )
      ALL = ALO(I)
      N1 = NODELM(1,I)
      N2 = NODELM(2,I)
      NK = NNODE
      TX1 =( X(N2) - X(N1) )
      TY1 =( Y(N2) - Y(N1) )
      TZ1 =( Z(N2) - Z(N1) )
      XP = X(NK) - X(N1)
      YP = Y(NK) - Y(N1)
      ZP = Z(NK) - Z(N1)
      AL(I) = SQRT ( TX1*TX1 + TY1*TY1 + TZ1*TZ1 )
      ALL = AL(I)
      AA = TX1*TX1 + TY1*TY1 + TZ1*TZ1
      AB = TX1*XP  + TY1*YP  + TZ1*ZP
      U1 = AA*XP - AB*TX1
```

```
      U2 = AA*YP - AB*TY1
      U3 = AA*ZP - AB*TZ1
      UU = U1*U1 + U2*U2 + U3*U3
      UU = SQRT(UU)
      CX = ( X(N2) - X(N1)  ) / AL(I)
      CY = ( Y(N2) - Y(N1)  ) / AL(I)
      CZ = ( Z(N2) - Z(N1)  ) / AL(I)
      C(1,I) = CX
      C(2,I) = CY
      C(3,I) = CZ
      C(4,I) = U1 / UU
      C(5,I) = U2 / UU
      C(6,I) = U3 / UU
      C(7,I) = C(2,I)*C(6,I) - C(3,I)*C(5,I)
      C(8,I) = C(3,I)*C(4,I) - C(1,I)*C(6,I)
      C(9,I) = C(1,I)*C(5,I) - C(2,I)*C(4,I)
      CALL SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,I,
     & NDPN )
      DO 51 J = 1 , 12
      II = LM(J)
      IF(II .NE. 0 ) THEN
      D2(J) = TOTLD(II)
      ELSE
      D2(J) = 0.0
      ENDIF
51    CONTINUE
      CALL ES3DBN(I,NELMNT,NNODE,AREA,E,AIX,AIY,AIZ,C,ALL,
     & NTD,G,ICHK,D3,D2,AXF,IW)
      DO 40 II = 1 , 6
      JJ = LM(II)
      MM = LM(II+6)
      IF(JJ . NE. 0 ) THEN
      PPRV(JJ) = PPRV(JJ) +  D3(II)
      ENDIF
      IF( MM .NE. 0 ) THEN
      PPRV(MM) = PPRV(MM) + D3(II+6)
      ENDIF
40    CONTINUE
30    End presched do
      Barrier
      End barrier
      Barrier
      IF( NNODE .LE.  10 ) THEN
      WRITE(IW,*)'              ==================== '
      WRITE(IW,*)'               TOTAL  U. B. F '
      WRITE(IW,*)'              ==================== '
      DO 72 II = 1 , NTD
      WRITE(IW,*)' P(',II,') = ',P(II)
72    CONTINUE
      ENDIF
      End barrier
      Barrier
      End barrier
      Critical TYPE20
      DO 31 JDJ = 1 , NTD
      P(JDJ) = P(JDJ)  - PPRV(JDJ)
31    CONTINUE
      End critical
      Barrier
      End barrier
      CCC = 1.00
      Barrier
      IF( NNODE .LE. 10 ) THEN
      WRITE(IW,*)'                 ================= '
      WRITE(IW,*)'                    U. B. F '
      WRITE(IW,*)'                 ================= '
      DO 71 IB = 1 , NTD
      WRITE(IW,*)' P(',IB,') = ',P(IB)
71    CONTINUE
      ENDIF
      End barrier
986   CONTINUE
      Barrier
      IF( METHOD .GT. -1 ) THEN
      UFNORM = 0.0
      DO 60 II = 1 , NTD
      UFNORM = UFNORM + P(II) * P(II)
      IF(UFNORM .GT. 1.0E20) GO TO 91
60    CONTINUE
91    UFNORM = SQRT(UFNORM)
      WRITE(IW,*)' UNBALANCED FORCE NORM = ',UFNORM
      IF( UFNORM .LE. TOL ) THEN
      DO 80 J = 1 , NTD
      P(J) = DELP(J)  + P(J)
80    CONTINUE
      JSTOP = 1
      ENDIF
      WRITE(IW,*)' UFNORM = ',UFNORM,'  TOL = ', TOL
```

```
      ENDIF
      End barrier
987   CONTINUE
      Barrier
      End barrier
      RETURN
      END
C **.*****************************************************************
C   This subroutine UBFITS is used to calculate the unbalanced forces *
C   in th truss element.                                               *
C **.*****************************************************************
      Forcesub UBFITS (IR,IW,NOMP,NELMNT,NNPE,YM,CTH,DM,AR,WIGHT,
     & NODELM,AL,CX,CY,CZ,X,Y,Z,MPN,ICHK,ALO,NUM4,MULT4,MID,AXF,ID,
     & NNODE,NDPN ,NTD,DELP ,P,NEWIT,HNORM,TOL,METHOD,JSTOP)
     & of NP ident ME
      REAL YM(1) , AR(1) ,CX(1),CY(1),CZ(1) , AL(1),ALO(1),AXF(1)
      REAL X(1) , Y(1) , Z(1) ,CTH(1),DM(1),WIGHT(1),DELP(1),P(1)
      Private REAL DCOL(3),AREA,E,TX,TY,TZ,PP
      Private INTEGER N1,N2
      Private REAL  PPRV(5000),PFNORM
      Shared REAL UFNORM
      INTEGER ID(6,NNODE),LM(24)
      INTEGER NODELM(NNPE,NELMNT),MPN(1),MID(1)
      Shared LOGICAL TYPE10
      End declarations
      DO 1 I = 1 , NTD
      PPRV(I) = 0.0
1     CONTINUE
      CONST1 = MULT4
      Presched do 20 J = 1 , NTD
      P(J) = DELP(J)  + DELP(J) * CONST1
20    End presched do
      Barrier
      End barrier
      Presched do 30 J = 1 , NELMNT
      AREA = AR( MID( J ) )
      E =    YM( MID( J ) )
      N1 = NODELM(1,J)
      N2 = NODELM(2,J)
      TX =( X(N2) - X(N1) ) **2
      TY =( Y(N2) - Y(N1) ) **2
      TZ =( Z(N2) - Z(N1) ) **2
      AL(J) = SQRT ( TX + TY + TZ )
      CX(J) = ( X(N2) - X(N1)  ) / AL(J)
      CY(J) = ( Y(N2) - Y(N1)  ) / AL(J)
      CZ(J) = ( Z(N2) - Z(N1)  ) / AL(J)
      AXF(J) = ( AL(J) - ALO(J) ) * E * AREA / ALO(J)
      PP = AXF(J)
      DCOL(1) = CX(J)
      DCOL(2) = CY(J)
      DCOL(3) = CZ(J)
      CALL SUBLM (IR,IW,NNODE,NELMNT,NNPE,ID,LM,NODELM,ICHK,J,
     & NDPN )
      DO 40 II = 1 , 3
      JJ = LM(II)
      MM = LM(II+3)
      IF(JJ . NE. 0 ) THEN
      PPRV(JJ) = PPRV(JJ) +  PP * DCOL(II)
      ENDIF
      IF( MM .NE. 0 ) THEN
      PPRV(MM) = PPRV(MM) - PP * DCOL(II)
      ENDIF
40    CONTINUE
30    End presched do
      UFNORM = 0.0
      Barrier
      End barrier
      Critical TYPE10
      DO 50 II = 1 , NTD
      P(II) = P(II) + PPRV(II)
50    CONTINUE
      End Critical
      Barrier
      End barrier
      IF(METHOD .GT. 1 ) THEN
      PFNORM = 0.0
      Presched do 60 II = 1 , NTD
      PFNORM = PFNORM + P(II) * P(II)
60    End presched do
      ENDIF
      Critical TYPE10
      UFNORM = UFNORM + PFNORM
      End Critical
      Barrier
      UFNORM = SQRT(UFNORM)
      IF(METHOD .GT. 1 ) THEN
      IF( UFNORM .LE. TOL ) THEN
      DO 80 J = 1 , NTD
```

```
      P(J) = DELP(J)  + P(J)
80     CONTINUE
      JSTOP = 1
      ENDIF
      ENDIF
      HNORM = UFNORM
      End barrier
      RETURN
      END
C **.*****************************************************************
C   This subroutine VECMUL is used to multiply a matrix time a vector.*
C **.*****************************************************************
      SUBROUTINE VECMUL(N1,N2,N3,A,B,C)
      REAL  A(N1,N2),B(N2),C(N1)
      DO 30 K = 1 , N1
      DO 20 I = 1 , N3
      SUM = 0.0
      DO 10 J = 1 , N2
      SUM = SUM + A(K,J) * B(J)
10    CONTINUE
      C(K) = SUM
20    CONTINUE
30    CONTINUE
      RETURN
      END
C **.*****************************************************************
C   This subroutine WRITEF is used to write out the output of the     *
C   F.E.A. part. (Displacement & Stresses).                           *
C **.*****************************************************************
                  S U B R O U T I N E       W R I T E F
(IR,IW,NELMNT,AXF,IPRINT,NPFLAG,TOTLD,NTD,ICHK,
     & NUM4)
      REAL AXF(1),TOTLD(1)
      IPRINT = 0
      IF( IPRINT .EQ. 1 ) THEN
      DO 10 J = 1 , NELMNT
      WRITE(IW,55) J , AXF(J)
10    CONTINUE
      ENDIF
      WRITE(11,*)NUM4,TOTLD(1)
      IF(NELMNT .LE. 20 ) THEN
      WRITE(IW,*)
      WRITE(IW,65)
      DO 20 J = 1 , NELMNT
      WRITE(IW,55) J,AXF(J)
20    CONTINUE
      WRITE(IW,*)
      WRITE(IW,*)
      write(6,105)(TOTLD(II),II=1,NTD)
      ELSE
      WRITE(IW,*)' DISP ( 1 ) = ',TOTLD(1)
      WRITE(IW,*)' DISP (', NTD/2 ,' ) = ',TOTLD(NTD/2)
      WRITE(IW,*)' DISP (', NTD   ,' ) = ',TOTLD(NTD)
      WRITE(IW,*)
      WRITE(IW,65)
      WRITE(IW,55) 1,AXF(1)
      WRITE(IW,55) 10,AXF(10)
      WRITE(IW,55) 20,AXF(20)
      WRITE(IW,*)
      ENDIF
55    FORMAT(/2X, I6 , E16.6 )
65    FORMAT(20X,' M E M B E R   F O R C E S '
     & /19X,    '============================' )
      RETURN
      END
```

VITA AUCTORES

Name: Majdi Ahmed Baddourah
Born: September 9, 1959.
Place of birth: Bierut, Lebanon

The author grew up in Kuwait receiving all his education up to the high school level in Hawalli city. He received his baccalaureate degree in the Civil Engineering from the University of South Carolina in May 1982. He also received his Master of Engineering from the University of South Carolina in May 1983. The author then worked in Kuwait for four years as a project manager of Al-Bayan school project. The author came to Old Dominion University in August, 1987 to continue his studies and was enrolled in the Ph.D. program in Civil Engineering. The author is Licensed as Professional Engineer in the Commonwealth of Virginia. He is a member of Chi Epsilon, American Institute of Steel Construction, National Society of Professional Engineers. The author has a few publications in the engineering field. The author is currently employed by Lockheed Engineering and Science Company as an Engineer.

PUBLICATION AND PRESENTATIONS:

1. Carmona, E.A., Baddourah M.A., Nguyen, D.T., Agarwal, T.K.," Parallel-Vector Solution for Nonlinear Finite Element Analysis," Final Report, Wl-TR-90-01, Weapons Laboratory, Air Force System Command, Kirtland Air Force Base, NM 87117-6008. pp 84, March 1990.

2. Baddourah, M.A., Nguyen, D.T., "Parallel-Vector Processing for Linear Programming," Journal of Computers and Structures, Volume 38, pp. 269-282 (1991).

3. Baddourah, M.A., Storaasli, O.O, Carmona,E.A., and Nguyen, D.T., A fast Parallel Algorithm for Generation and Assembly of Finite Element Stiffness and Mass Matrices," Presented at the AIAA/ASME/ASCE/AHS 32nd SDM Conference, Baltimore, Maryland(April 8-10,1991)

4. Nguyen, D.T., Storaasli, O.O, Carmona, E.A., Al-Nasra, M., Baddourah, M.A. and Agarwal, T.K., "Parallel-Vector Computation For Linear Structural Analysis and Nonlinear Unconstrained Optimization Problem," to appear in Computing System in Engineering, An International Journal (Pergmon Press).